PV251 Visualization

Autumn 2024

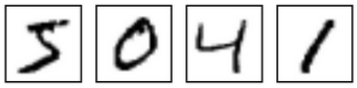
Study material

Lecture 7: Multivariate data visualization

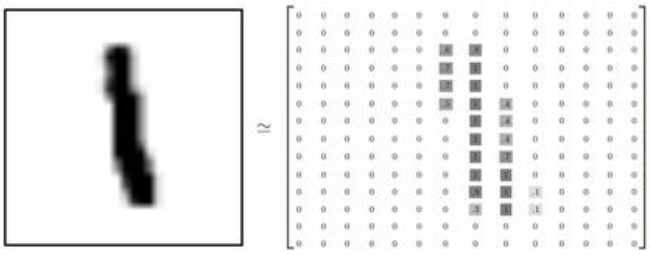
We refer to multivariate (or high-dimensional) data as data that consists of different types of attributes. As an example, we can mention a data set, which is created by collecting information about the weight w, height h and number of shoes from a random sample of people. Then the triplets (w_1, h_1, s_1), (w_2, h_2, s_2)… are an example of a multivariate data set (or multivariate data).

In this lecture, we will deal with techniques for visualizing high-dimensional data that generally does not have explicit spatial attributes.

Among other examples of this type of data belongs the MNIST dataset that consists of 10,000 hand-written digits.

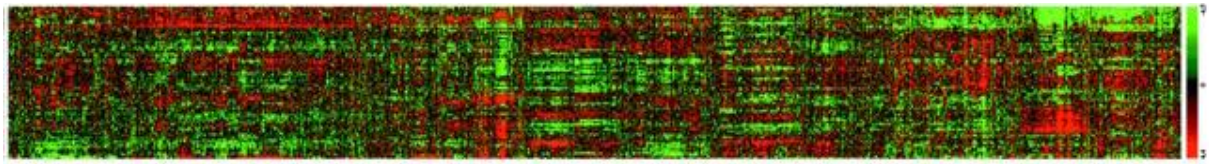


Each image containing one digit has size of 28x28 pixels. This leads to the 784-dimensional feature vector, where each dimension stands of one intensity value in the image.



The goal of the analysis of this dataset is to automatically determine the digit on a particular image. This consists of the task to classify the dataset into bins according to the number on the image. And for this, we can use techniques for the analysis and visualization of high-dimensional datasets.

Another typical example is the gene expression dataset. In this case, genes are representing dimensions and experiments are individual samples. This technique measures the expression of genes in different experimental settings (conditions, species). This often results in few samples but many genes.



Curse of Dimensionality

Multidimensional datasets and their exploration suffer from several problems, tightly related to the dimensionality. The efficiency of many algorithms tailored to the exploration of such datasets depends on the number of dimensions these datasets possess. Another problem is that, very often, when the number of dimensions is increasing, the data itself becomes sparse, i.e., the values in many dimensions are missing. Also, when we are using these datasets for machine learning algorithms for automatic information retrieval, the number of training samples is exponentially growing with the increasing number of dimensions. Visualization aims to address these issues by providing the users with visual insight into the outcomes of these algorithms. Visualization aids namely in the tasks related to the visual exploration of high-dimensional data, such as detecting clusters, finding regularities and irregularities, or identifying relevant data dimensions. Visualization then enables the visual inspection of the classification results. In consequence, this should enable us to better understand and assess the quality of the outcomes of given algorithms.

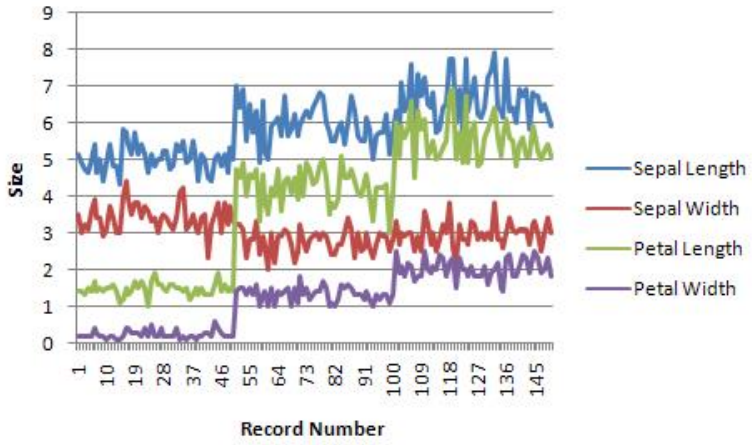
In the following, we will be often operating with the exemplary Iris dataset that is quite often used for demonstrating the capabilities of the multidimensional visualization techniques. It consists of 3 different iris species and contains 50 samples for each of them. There are four measured features: length and width of sepals and petals.

Now we will be presenting a set of techniques that are typically used for the visualization of multidimensional data. We will start with the line-based techniques.

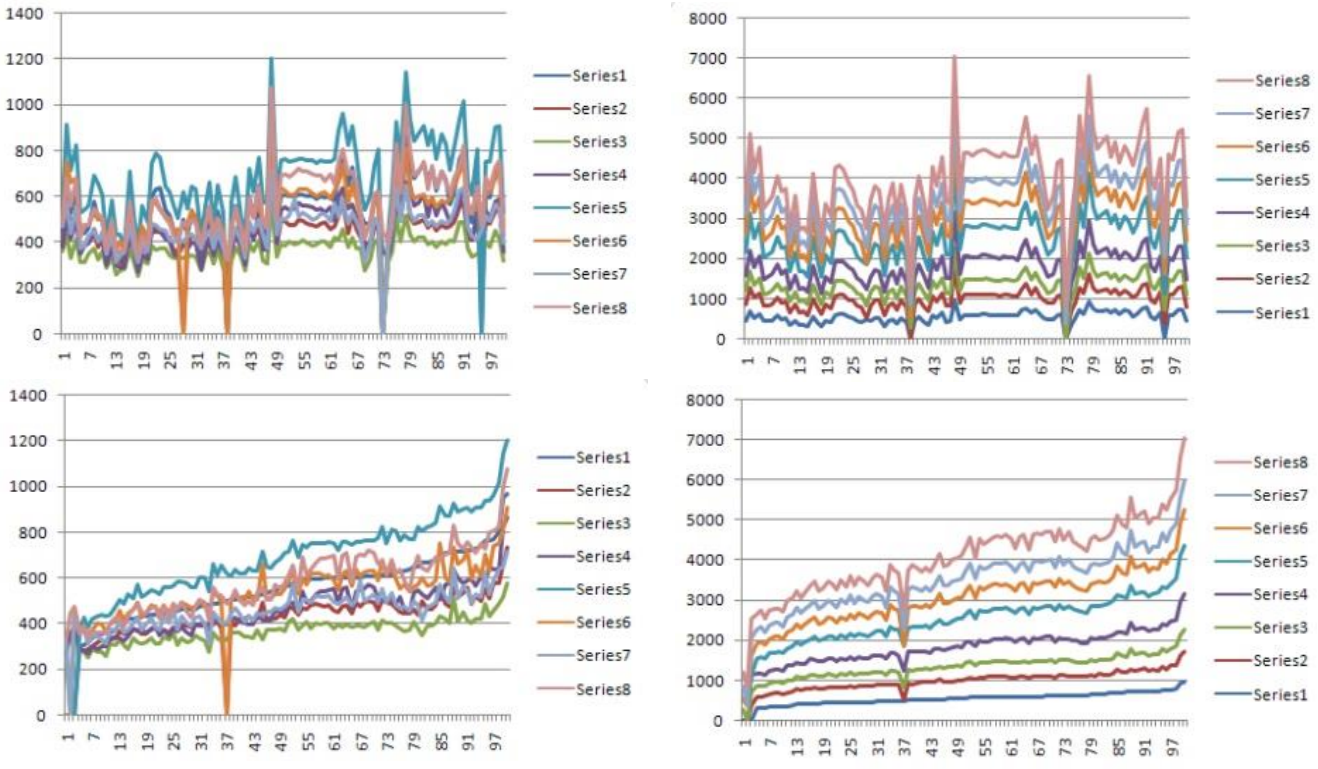
Methods for visualizing point data represented each record using a marker. Line-based techniques display records by connecting the corresponding points with a straight or curved line. These lines not only emphasize the relationships between data values, but also convey other perceptible properties through various chamfers, curvatures, crossings, and other characteristics of line patterns.

A line chart is a visualization technique with one variable, in which the vertical axis represents a possible range of values of the variables and the horizontal axis represents a certain arrangement of records in a given data set.

Most techniques for displaying a single variable can be extended to multiple variables (multivariate data) – using the already known superimposition or juxtaposition techniques. For line charts, we can plot data for a reasonable number of dimensions using a common set of axes. Other dimensions are distinguished by color, type of drawn line, its width, or other graphic attributes (see figure).



When increasing the number of dimensions, or if there is already a large overlap of data, the use of the technique of superimpositioning becomes problematic. The following figure at the top left shows an 8-dimensional dataset (salaries at faculties for different functions at 100 different universities). It is obvious that in this classical display using superimpositioning it is difficult to know the data. However, some strategies can be used to improve the interpretation. The image on the top right is the so-called stacked line chart, where instead of using a common basis for plotting, the graph of the previous dimension is used as the basis for each additional dimension.



The pictures on the left and bottom right show the use of a different strategy – sorting records by one dimension.

The effectiveness of the above examples depends mostly on the fact that the dimensions have common units in the axes. If the individual variables (corresponding to the dimensions) have different units, the situation becomes much more complicated. One common approach is to use multiple vertical axes, each labeled separately. Another possible approach is to create a set of graphs, one for each dimension. We then stack these graphs vertically (usually after applying scaling in the vertical dimension so that most graphs can be displayed at the same time).

RadViz technique

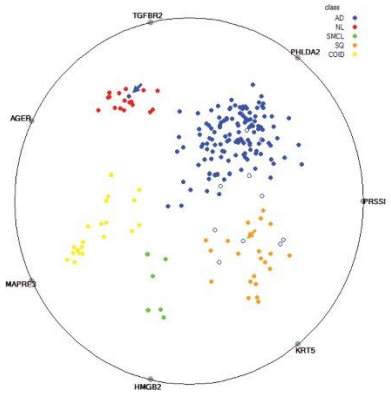
Another technique for the visualization of multidimensional data is the force-driven technique for distributing points called RadViz. It is based on physics, more precisely on Hooke's law, and uses the finding of the equilibrium position of a point for the representation. For an N-dimensional data set, N "anchor" points are placed on the circumference of the circle, which represent the fixed ends of the N strings assigned to each data point. To simplify the calculation and provide an intuitive view of this algorithm, we place the anchors on a circle with a radius of 1.0, the center of which is at the beginning of the coordinate system.

Thus, for a given normalized data vector $D_i = (d_{i,0}, d_{i,1}, ..., d_{i,N-1})$ and a set of unit vectors A, where A_j represents the j-th anchor point, we get the following equilibrium calculation:

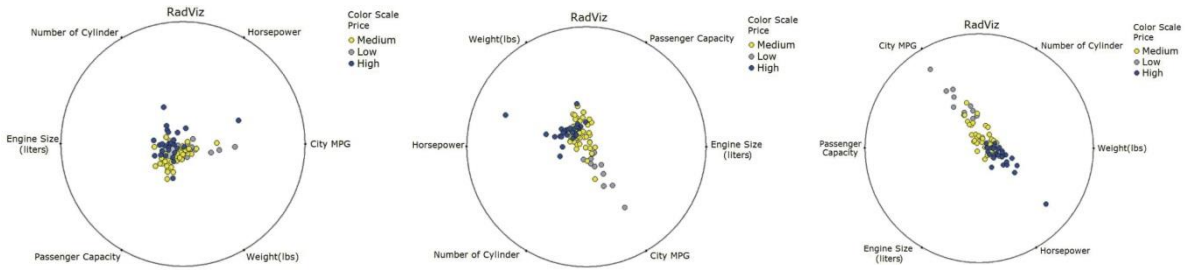
$$\sum_{j=0}^{N-1}(A_j - p)d_j = 0$$

Where *p* is the vector for the point in equilibrium. The calculation of *p* is performed according to the formula:

$$p = \frac{\sum_{j=0}^{N-1}(A_j d_j)}{\sum_{j=0}^{N-1} d_j}$$



Note that different placement and arrangement of anchors leads to different results, and that points that are different in N dimensions can be mapped to the same location in 2D. However, this is a problem with all projection and dimension reduction techniques. In the case of RadViz, a simple solution is to enable interaction, such as allowing movement with anchors and observing changes in visualization. In this way, it is often possible to trace relationships in the data, as shown in the following figure.

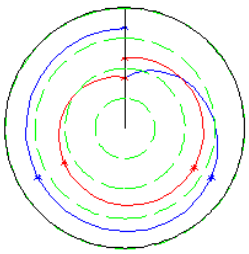


The figures show different views of the same dataset displayed by the RadViz technique. The figures show the results after manually rearranging the individual dimensions (by moving the anchors). In addition, cars are painted according to their price. The goal is to find the attributes of cars that best predict which price category a car will fall into based on the given attributes.

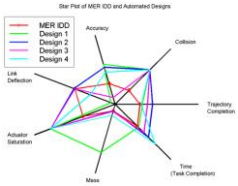
It is necessary to keep in mind that this is a lossy transformation.

Other radial-based techniques

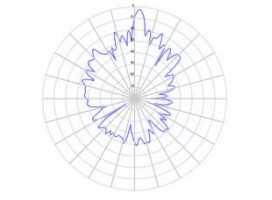
For each technique that has a coordinate system oriented horizontally and / or vertically, there is an equivalent technique using radial orientation. For example, a **radial line chart** is a chart where the drawn lines represent the offset from the circular base (see figure).



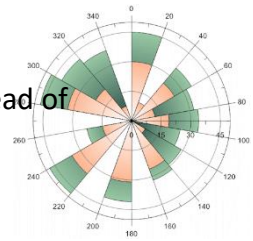
A large graph can be transformed by dividing it into equal segments and mapping each segment to a base with a different radius. This is especially useful for studying cyclical events. Individual variants of pie charts also include **radar** and **star charts**.



In addition to these popular techniques, the number of other radial-based charts have been developed, such as:



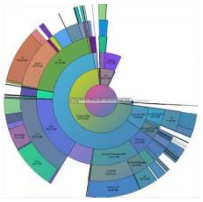
- **Polar charts** - graphs showing polar coordinates



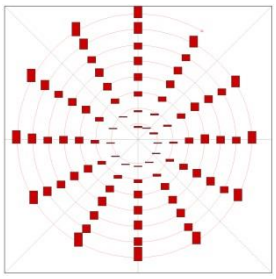
- **Radial stacked bar charts** - similar to radial line charts, only bars are displayed instead of lines

- **Radial area charts** - similar to line charts, the area below the lines is additionally filled with color or texture

- **Radial bar charts** - columns are represented by circular arcs with a common center. The difference between a pie chart and a graph is that in one the bar is straight, and the base is curved, while in the other it is exactly the opposite.



All techniques using radial axes and involving more than one circle use either concentric circles or a continuous spiral. The following figure is given as an example of a bar graph with a spiral base.



This method, unlike concentric circles, does not show discontinuities at the end of each cycle. The comparison within and between cycles is relatively simple, especially when the columns are oriented along a vertical axis (as in the figure) instead of a perpendicular orientation to the spiral.

Thanks to the knowledge of human perception, we know that in this case it is more difficult to measure the difference between adjacent elements than when using a common base (traditional bar graph). However, the traditional column expression does not allow us to simply follow the patterns between the individual elements in the same position in different cycles.

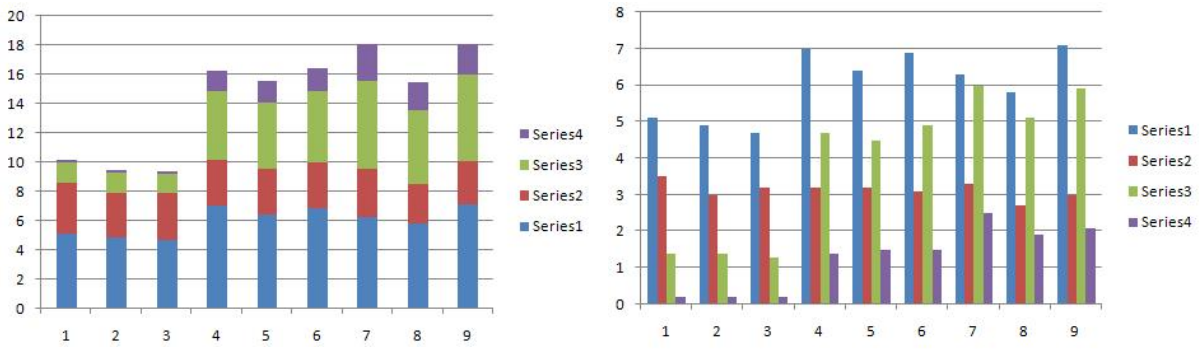
Techniques for area data

For area data techniques, filled polygons of a given size, shape, color, and other attributes are used to display values. Although we know from human perception errors that our ability to accurately interpret an area is inferior to our ability to measure other attributes, such as length, several very effective techniques have been developed to display data in this category. The goal of some of these techniques is not to show the raw data themselves, but their clusters or distributions of values.

Many of these techniques for visualizing area data were originally designed for univariate data, but some of them have been expanded by more dimensions. We will now introduce some of these methods.

Stacked bar charts

If we display multivariate data, we have several options for using bar charts. A common technique is a layered bar graph, where each column consists of several shorter columns representing the values in each dimension. Color, texture, and others are commonly used to distinguish them (see the picture on the left).



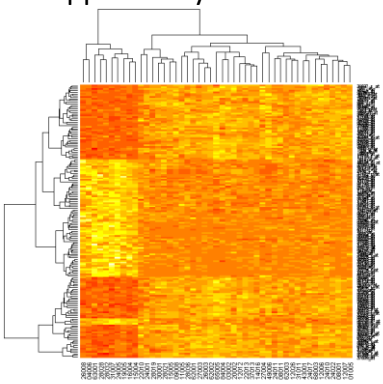
A similar approach shows columns for individual variables right next to each other (see figure on the right). The columns therefore have a common base, which simplifies their interpretation.

The choice between these two approaches often depends on the number of variables and the number of columns. Stacked columns do not require additional horizontal space requirements, while the second technique of "adjacent" columns may require much more space in this direction.

Tabular visualizations

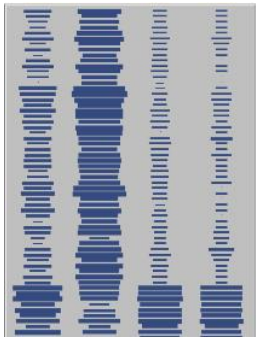
Multivariate data is often stored in tables, so several visualization techniques have been developed that work with these structures. These techniques usually differ in the type of interactions they support. One example is the so-called **heatmaps**. They are created by displaying a table of records using color instead of text. In this visualization technique, all data values are mapped to the same normalized color space, and each value is rendered as a

colored square or rectangle. The use of different color maps, along with allowing the user to expand or reduce colors to emphasize or suppress certain ranges of values, significantly increases the applicability of this technique.



Permutations or refoldable grids are basically heatmaps that allow you to reorganize rows and columns to reveal certain data properties. Columns and rows can be reorganized to maximize diagonalization - creating a matrix with cells aligned along the main diagonal. Other variants rearrange the data to isolate clusters with similar values or patterns in the data values.

Another technique is the so-called **survey plots**. They are a variant of the permutation matrix, where instead of staining the cells, we work with their size. In addition, we align the centers of the cells to the individual attributes. This alleviates color perception errors caused by various side effects of neighboring colors.



However, because area measurement is much more error prone than length measurement, this method also has its errors.

The figure shows a survey plot calculated using the DataLab tool. Each column is a visual representation of one of the four dimensions of the iris dataset.

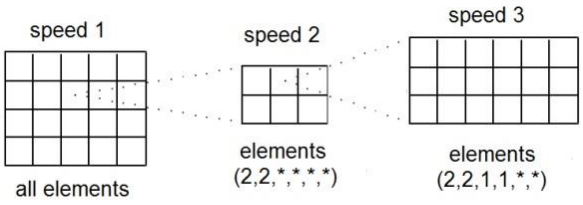
Dimensional stacking

The technique of folding dimensions was developed by LeBlanc et al. and aims to map data from discrete N-dimensional space to a 2D image in such a way as to minimize data occlusion while preserving most of the spatial information.

Briefly, the mapping is performed as follows:

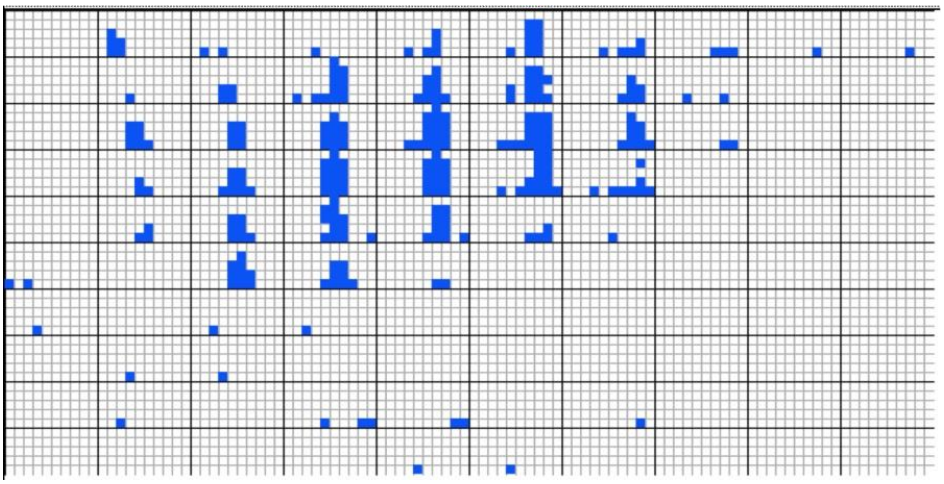
- We start with data of dimension 2N + 1 (for an even number of dimensions it is necessary to supply an additional default dimension of cardinality 1).
- Select the final cardinality for each dimension.
- Select one of the dimensions as a dependent variable. The rest are considered independent variables.
- Now we create ordered pairs of independent variables (N pairs) and assign each pair its unique value (called velocity) from 1 to N. The pair corresponding to velocity 1 creates a virtual image whose size corresponds to the cardinality of dimensions (the first dimension of the pair is oriented horizontally, second vertically). At each position of this virtual image, another virtual image is created that corresponds to the dimensions of velocity 2. Again, the size of this image depends on the cardinality of the corresponding dimensions. This process is repeated until all dimensions are included. In this way, we achieve that each location in a space with many dimensions has its own unique location in the 2D image that results from the mapping.

The value of the dependent variable at a given location in a multidimensional space is then mapped to the color / intensity of that location in the 2D image. The whole process is illustrated by a picture. A 6-dimensional data set is shown, where dimensions d_1 to d_6 have cardinalities 4, 5, 2, 3, 3, and 6.



In other words, this technique works as follows. It starts by discretizing the ranges in each dimension. Each dimension is then assigned an orientation and arrangement. The dimensions with the two lowest configurations are used to divide the virtual screen into sections, with the cardinality of the dimensions determining how many sections in the horizontal and vertical axes are generated. The next section created in this way is used to recursively define the virtual screen in the other two dimensions in the same way. This process is repeated until all dimensions have been processed and the data has been placed at its position on the screen.

As an example, let's take 4D data visualized by dimensional stacking.



The composition of dimensions can be displayed using an N-dimensional histogram if the color of the cell is set in proportion to the data values that are mapped to it.

Worlds-within-worlds methods or treemaps are based on a similar principle, which we will discuss in future lectures.

Other techniques

Glyphs and icons

In the context of data and information visualization, a glyph is a visual representation of a piece of data or information where the graphic entity and its attributes are controlled by one or more attributes of the input data. For example, the width and height of a block can be controlled by the student's results in the semester and at the end of the year, while color can be associated with the student's gender.

The definition is very general because it includes glyphs used in scatterplots, histogram columns, or even entire lines in a graph.

Many authors have developed lists of graphical attributes to which data values can be mapped. These include position (1D, 2D, 3D), size (length, area, volume), shape, orientation, material (brightness, saturation, intensity, texture, transparency), line style (width, dashes, convergence) and dynamics (speed of motion, direction of motion, blinking rate).

We will now list a number of possible mappings to different types of glyphs, including:

- **1: 1 mapping**, where each data attribute is mapped to unique and different graphical attributes
- **1: many mapping** where a set of redundant mappings is used to achieve greater accuracy and simplicity of interpretation

- **Many to many mapping** where several or all data attributes are mapped to a common type of graphical attribute, separated in space, orientation, or other type of transformation

1: 1 mapping is often designed to take advantage of user knowledge - using intuitive data pairing to graphical attributes to simplify the comprehension process. Examples include mapping color to temperature or mapping flow direction to line orientation.

Redundant mapping can be useful in situations where the number of dimensions of input data is low, and the goal is to minimize the possibility of misinterpretation. An example is the mapping of a population to size and color at the same time, which will make the analysis accessible to people with color perception disorders, and in addition we will refine the comparison of two populations with similar values.

1 : many mappings are most advantageous in situations where it is important not only to compare the values of different records in the same dimension, but also to compare different dimensions of the same record. An example is mapping each dimension to the height of a vertical column to allow both comparisons inside and between records.

The following list contains a subset of glyphs that have been designed in various studies and are commonly used. Some are designed specifically for certain applications, such as flow visualization, while others are general.

- Profiles - height and color of columns

- Star-shaped glyphs - the length of evenly spaced rays emanating from the center

- Metroglyphs - length of rays

- "Stick" images - length, angle, and color of branches

- Trees - length, thickness, angles, and branches; the structure of the branches is derived from the analysis of the relationships between the dimensions

- Autoglyphs - color of cubes

- Blocks - height, width, depth of the first block + height of the following blocks

- Hedgehogs - "thorns" of a vector field with different orientation, thickness, and taper

- Faces - size and position of eyes, nose, mouth, curvature of mouth, angle of eyebrows

- Arrows - length, width, taper, base line color and the arrows themselves

- Polygons - highlighting local deformations in a vector field using changes in orientation and shape

- Dashtubes - texture and transparency for displaying vector field data

- Weathervanes

- Circular profiles - distance from the center to the vertices at the same angles

- Colored glyphs - colored lines across the cube

- Bugs (beetles) - the shape of the wings controlled over time, the length of the antennae, the size and color of the body, the size of the marks on the body

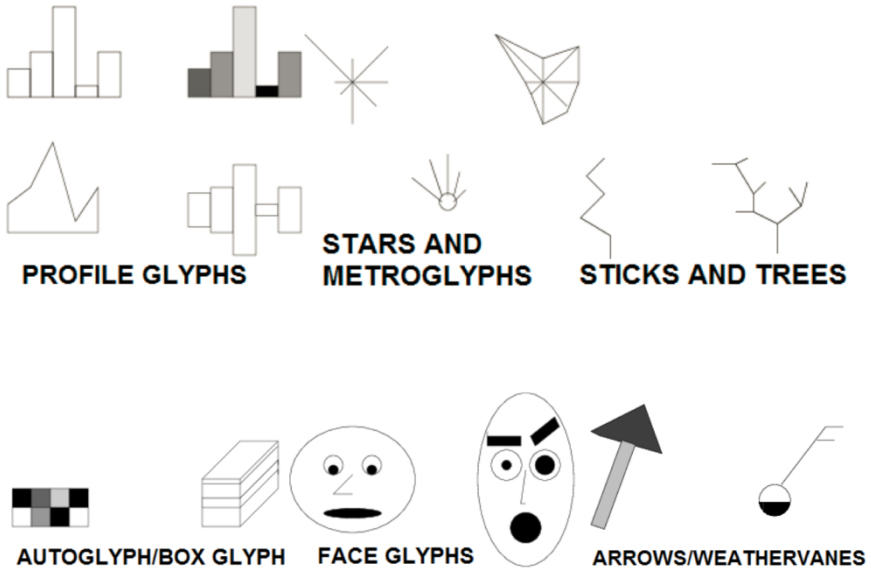
- Wheels - The 3D wheel maps time to height, the value of the variable to radius

- Boids - the shape and orientation of primitives moving in a time-varying field

- Procedural shapes - "blobby" objects controlled by up to 14 dimensions

- Glyphmaker - user controlled mapping

- Icon Modeling Language - attributes of a 2D contour and parameters that extract it into 3D and further transform or deform it



When using glyphs in the field of information visualization, we must be aware of a number of inaccuracies and limitations of this technique. The most important are the inaccuracies in perception, which depend on what graphic attributes we used. Some attributes, such as line length, can be judged much more accurately than others, such as orientation or color.

Other sources of inaccuracy stem from, for example, the fact that relationships between adjacent graphical attributes are much better interpreted than those that are at a greater distance. A comparison of two glyphs works in a similar way. If they are placed close to each other on the screen, they are easier to compare than when they are at a greater distance.

Finally, the number of data and record dimensions that can be efficiently displayed using glyphs is limited.

If we have already selected the type of glyph we want to use, there is N! different dimensional arrangements that can be used in mapping. There are several strategies for choosing the right layout:

- Dimensions can be sorted based on their correlation - similar dimensions are mapped to adjacent values. This helps to reveal general trends in the data.
- Dimensions can be mapped in such a way that we increase the influence of glyphs with a symmetrical shape, which are easier to perceive and remember. Shapes that are less symmetrical than their neighbors also dominate.
- Dimensions can be sorted according to their values in one record. For example, if the data contains multivariate time slots, then sorting based on the first record can highlight trends over time, seeing which relationships between dimensions are permanent or, conversely, change significantly over time.
- Dimensions can be sorted manually based on the user's knowledge of the domain. Semantically similar dimensions can be clustered to simplify interpretation.

The last important consideration when designing a glyph visualization is the placement of the glyphs on the screen. There are three basic types of deployment strategies:

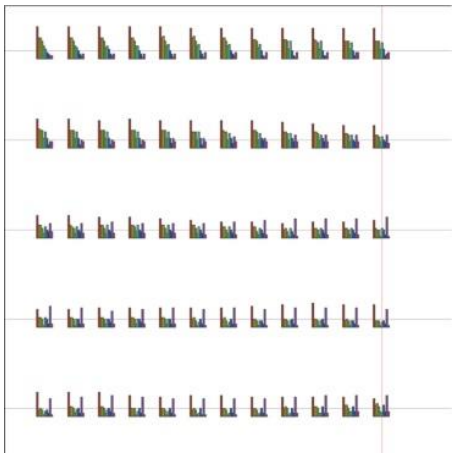
1. Uniform

2. Controlled data

3. Structured controlled

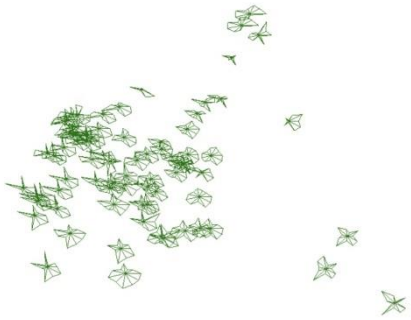
Uniform placement

Glyphs are scaled and distributed evenly across the screen (same gaps between glyphs). This strategy eliminates overlaps while making efficient use of screen space. Different record classifications reveal different data properties (see figure).



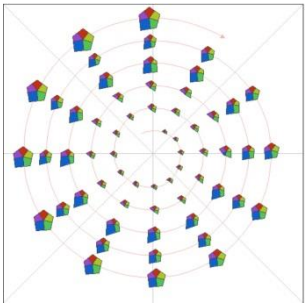
Data-driven placement

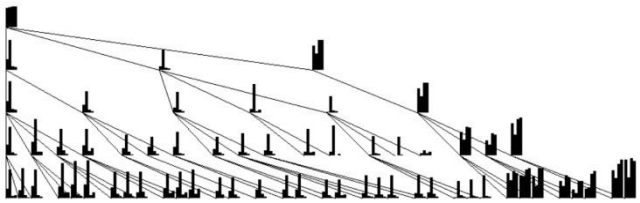
Data values are used to control glyph placement. There are two possible approaches. In the first, two (three for 3D viewing) dimensions are selected that control the deployment. In the second approach, positions are derived using algorithms such as PCA and MDS.



Structure-driven placement

If the data has an implicit or explicit structure, such as a cyclic or hierarchical one, this information can be used to control the distribution of the data. For example, glyphs can be arranged in a spiral or grid.





Dense Pixel Displays

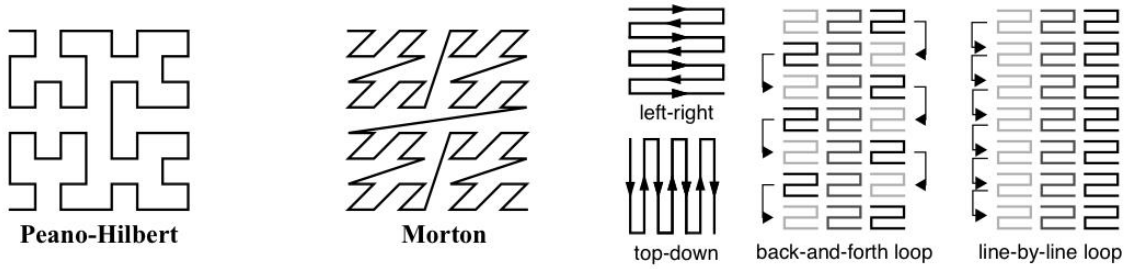
Dense pixel displays, also known as pixel-oriented techniques, are a hybrid method at the interface of point and regional (area) methods. The technique was developed by Keim and his colleagues and maps each value to individual pixels and creates a filled polygon for each dimension. These display types make the most of screen space, allowing millions of values to be displayed on a single screen. Each data value controls the color of one pixel - by changing the color map used, we can potentially reveal new data properties. Given the input data set and color map, it is still necessary to resolve the layout of data records and their arrangement.

In its simplest form, each dimension of a dataset generates a separate "subpicture" on the screen. Thus, we can consider each dimension as an independent set of numbers, each controlling the color of the corresponding pixels. Then it is necessary to arrange the elements in these sets in such a way that we emphasize the relationships between points

that are close to each other in the set. For example, we create a subpicture, where we alternate the passage from left to right and from right to left, and if we reach the edge of the subpicture, we move one row lower.

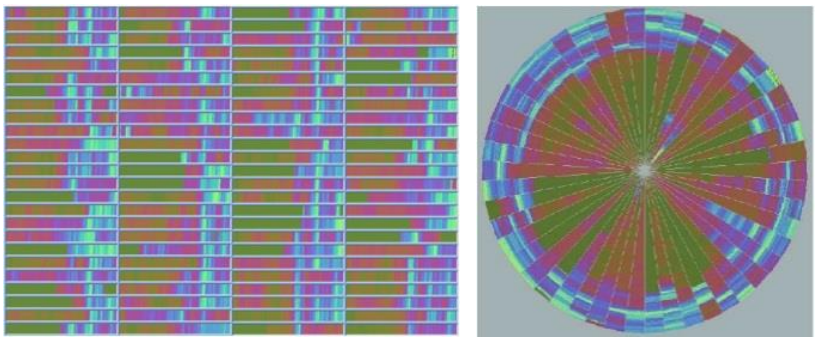
Another option is to use a spiral layout, where the first data point is placed in the center of the subpicture and the subsequent points are arranged in concentric squares.

There are a number of different ways to deploy, some of which are shown in the pictures.



Subpictures corresponding to the data for each dimension can be placed on the screen in different ways. The easiest way is to create a grid of sub-images that maximizes screen usage. Grids can have different configurations based on the arrangement of dimensions, which makes it possible to reveal correlations between dimensions. A technique called **recursive patterns** uses a grid pattern of subpictures.

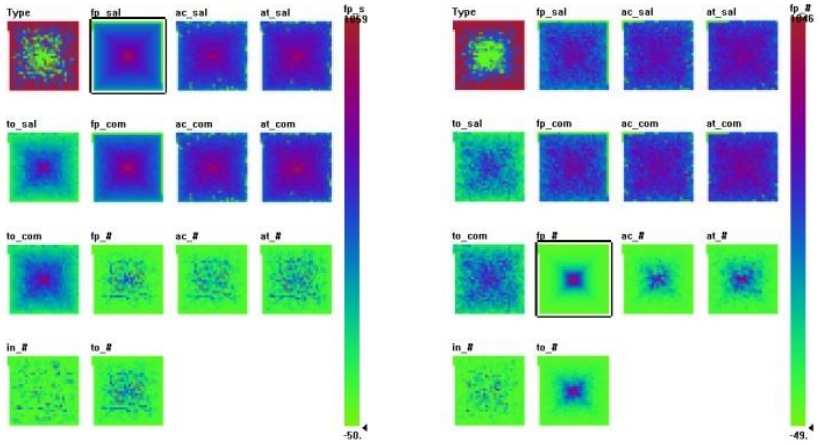
Another variant is the so-called **circle segments**, where instead of placing the pixels in rectangular subpictures, we place the pixels in circular "wedges". We start in the middle of the circle and intertwine back and forth from the center. Each dimension occupies the Nth arc of a circle, where N is the number of dimensions.



The last important topic in the design of pixel-oriented displays is the arrangement of data. For some types of data, such as time sequences, the arrangement is predetermined and fixed. However, in other cases, reordering records can reveal many interesting features. For example, if the data is organized based on one of the dimensions, clusters of values appear in that dimension. The same is true for other dimensions.

Another possible approach is to arrange the records based on their N-dimensional distance from the selected point.

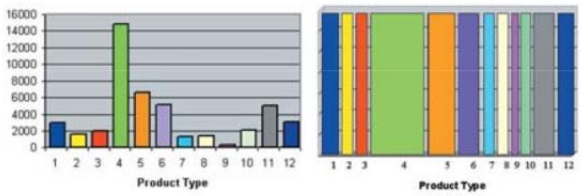
The figure shows the same data, which differs in the arrangement of the records.



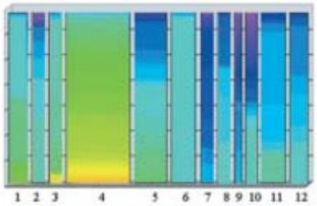
Pixel bar charts

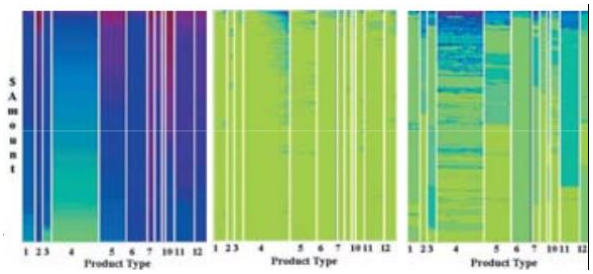
Dense pixels can also be placed in a standard bar chart. To make efficient use of screen space, pixel bar diagrams often use the width of the column instead of its height to represent aggregated data parameters. In addition, the columns are colored pixel by pixel so that we can display detailed information about the individual values of the data aggregated in the columns.

- Classic bar chart overload - include more information about individual elements



- Each pixel of a column corresponds to a data point belonging to the group represented by that column





The example shows the relationship between product type and price. The color is mapped to:

a) Amount spent

b) Number of visits

c) Sales volume

The next image shows several pixel bar charts that use the same pixel arrangement inside the columns (broken down by month, sorted on the y-axis by number of purchases, and on the x-axis by number of visits). Visualization allows the user to observe interesting facts about transactions, such as:

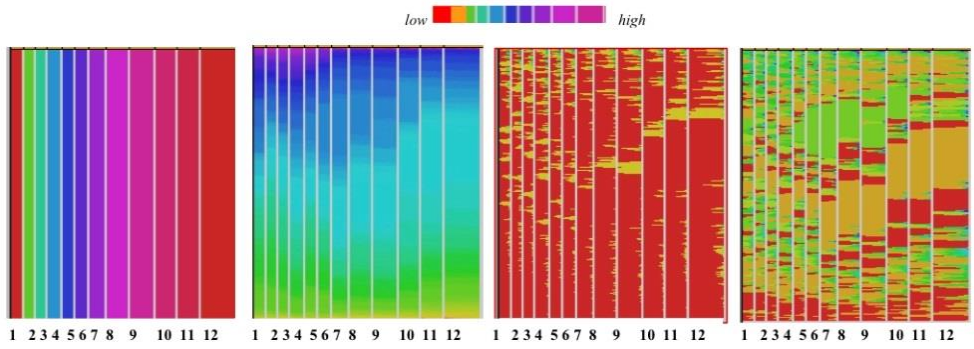
- In December there was the largest number of customers, while in February, March and May there were the fewest

- From February to May was the largest number of purchases

- The number of purchases in December is average

- From March to June, customers returned more often than in other months. December customers were mostly one-time.

- Customers who buy the most return more often and buy more things.



a) Color: Month b) Color: Purchase Amount c) Color: # of Visits d) Color: Quantity

Scatterplots

One of the most used visual representations of the multidimensional datasets are scatterplots. In this context, scatterplots can be thought of as a type of visualization that projects records from n-dimensional data space into any k-dimensional space of the output device (e.g., display), where data records are mapped to k-dimensional points. Each record is associated with a specific graphical representation (tag). Scatterplots can display individual records or summary records and can be structured based on the use of various projection techniques.

We have already encountered scatterplots several times, mainly because scatterplots are one of the first and most widespread visualization techniques used in data analysis. Most information analysis tools and packages contain some form of 2D and 3D scatterplots. Their success stems from our natural ability to estimate relative positions within a limited space. With the increasing dimensionality of the input data, the visual analysis consists of:

- **Multiple displays** - display of several graphs, each displaying some of the dimensions (display using superimposition or juxtaposition).

- **Dimension subsetting**, where we allow the user to select only a certain subset of input dimensions to be displayed. When searching for a suitable subset of dimensions, we can use algorithms to search for dimensions that contain the most useful information for a given task.

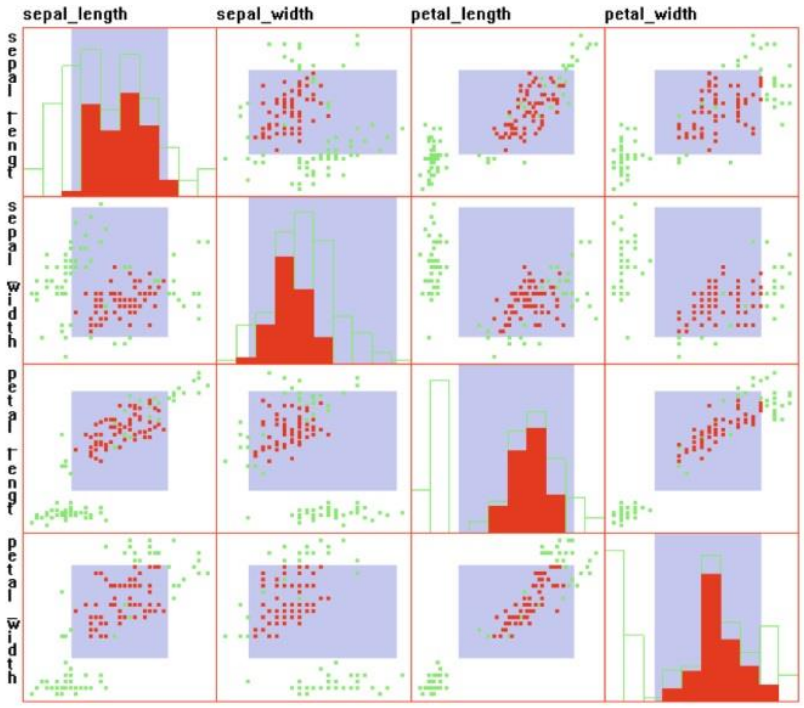
- **Dimension reduction** - using techniques such as PCA (principal component analysis) or multidimensional scaling, which allow you to transform data on higher dimensions into data on lower dimensions, while trying to preserve as many of the original relationships between data points.

- **Dimension embedding** - mapping of dimensions to other graphic attributes in addition to position, such as color, size, and shape (of course, the number of dimensions that can be displayed in this way is limited).

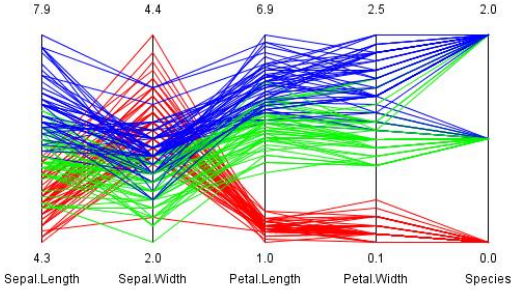
Withing this lecture, we will be discussing these approaches in more detail.

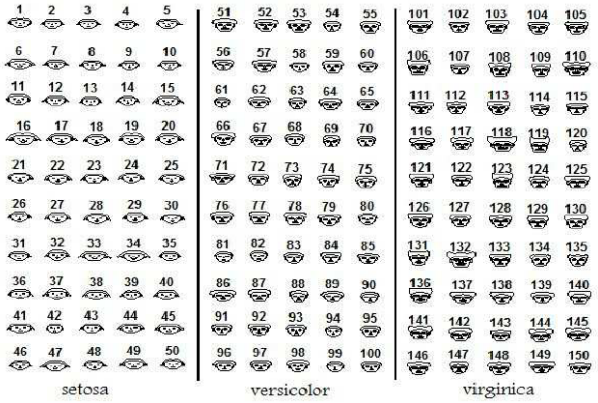
Multiple displays

In the case of displaying several graphs showing different dimensions of the displayed data (multiple display), the most frequently used technique is the so-called **scatterplot matrix**. It consists of a grid containing scatterplots that has N^2 cells, where N is the number of dimensions. Therefore, each pair of dimensions is drawn twice – it differs only by rotating the graph by 90 degrees. The arrangement of the dimensions is usually the same in the horizontal and vertical axes, which leads to the symmetry of the matrix along the main diagonal. Graphs on the main diagonal, which should show a variable in a given dimension with itself, are often used to communicate dimension information in the corresponding row / column or to plot a histogram of that dimension.

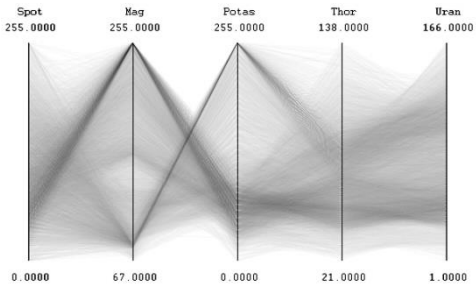


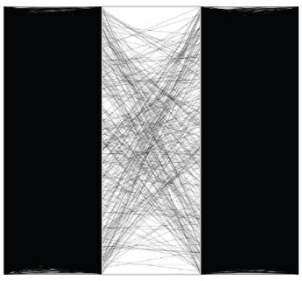
Among other possible representations belong, for example, parallel coordinates or Chernoff faces.

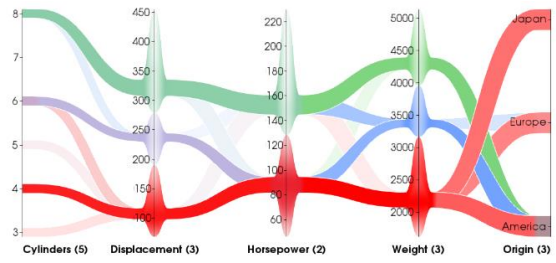




However, all these techniques are suffering from the scalability problem regarding the number of data items that can be comprehensively displayed. This problem can be partially solved by using different techniques, such as density-based visualizations, random sampling of data items, or edge bundling.







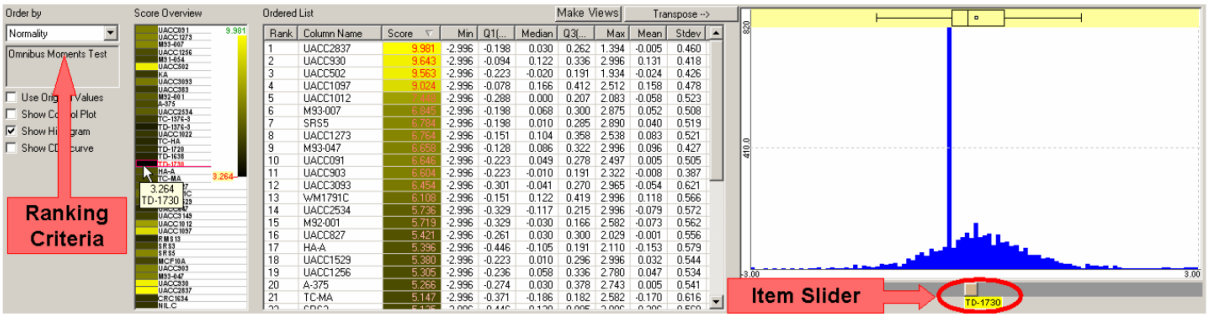
However, another possible problem is the number of dimensions in the dataset, that cannot be solved by these techniques. In the following, we will be discussing potential solutions of this problem.

Feature selection (dimension subsetting)

These techniques are based on selecting a subset of existing features of the dataset without any transformation. On other words, we are not applying any algorithm for dimension reduction, instead we are exploring only a selected subset of the original features. For these tasks, we are utilizing the multidimensional data visualization techniques that we've already seen or others, such as ranking techniques. These are ranking individual dimensions or dimension pairs based on a quality metric. Among typical quality metrics belong, for example, the number of outliers, correlation between the pairs of dimensions, image-based descriptors, etc. These quality metrics can be combined as well.

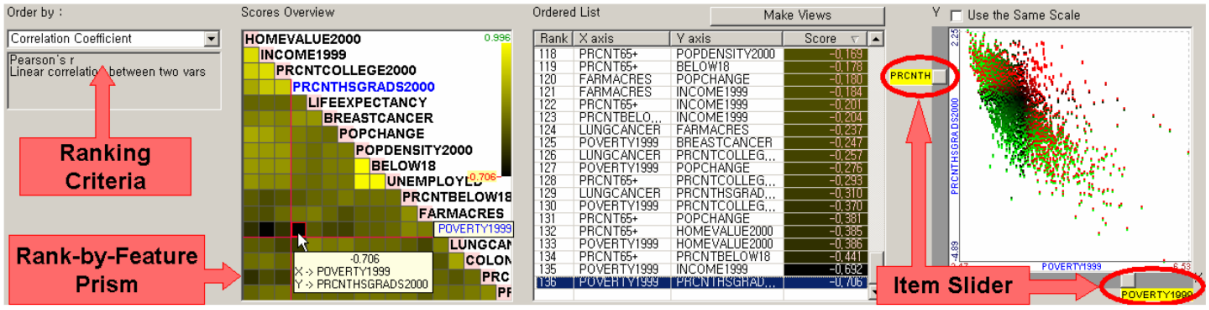
An example of the ranking approach is the following framework for the exploratory analysis of multidimensional data. It supports 1D and 2D ranking criteria:

1D criteria are the normality or uniformity (entropy) of distribution, the number of potential outliers, and the number of unique values.

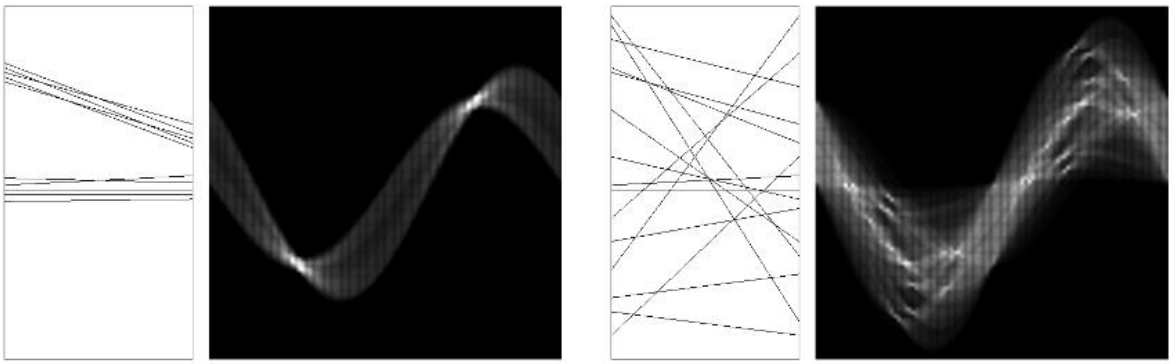


Here the individual views are interactively linked, and the histogram is ordered according to current ranking criteria.

2D criteria are the correlation coefficient, the least squares error for linear regression or curvilinear regression, the number of items in the region of interest, and the uniformity of scatterplots.

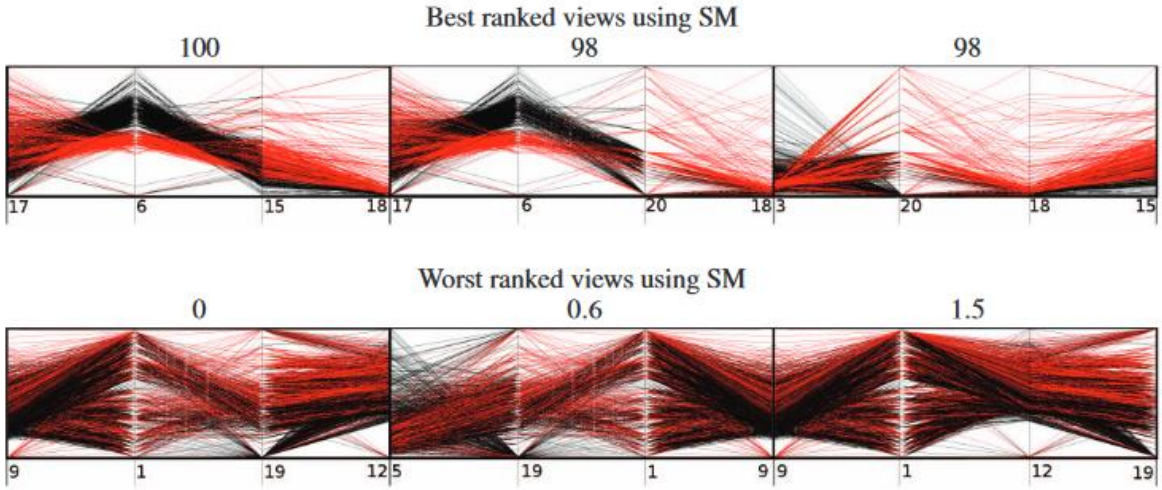


Another typical example of the feature selection technique is an approach to solving the problem of the selection and ordering of parallel coordinate axes. Here a possible approach would be to convert every dimension pair to the Hough space.



The main idea of the Hough transform is to define a straight line according to its parameters, i.e., the slope *a*, and the interception *b*. The image shows two examples of parallel coordinates and their respective Hough spaces. The left image presents two well defined line clusters and is more interesting for the cluster identification task than the right image, where no line cluster can be identified.

The application of this method can be demonstrated on the following example showing the dataset of cars. It consists of 7,404 cars, each having 24 attributes. The following image shows results of two rankings, where the criterion was the benzine (black) and diesel (red) engine.



Feature extraction (dimension reduction)

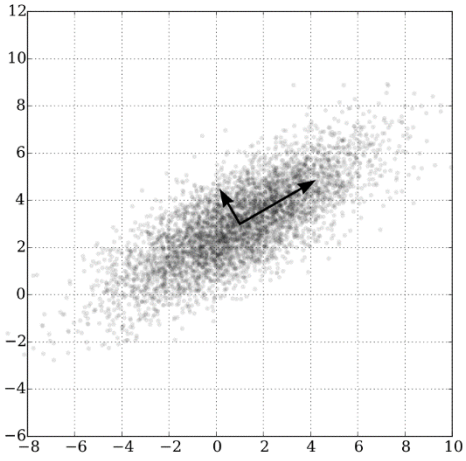
These techniques are actually transforming the existing features into lower dimensional space that is then displayed and explored by the user. For that, we can use the traditional and mostly already mentioned 1D / 2D /3D / nD visualization techniques.

The techniques for dimension reduction can be divided into two main categories – linear and non-linear projections.

Linear projections

These techniques are based on the linear transformation that projects the data from high-dimensional space to the low-dimensional space. The typical representatives of this type of projections are the Principal Component Analysis (PCA) or Multi-Dimensional Scaling (MDS).

PCA is projecting the data to the lower dimensions – so called principal components. The first principal component aims to capture as much variability of the data as possible. The other principal components are orthogonal to the principal one.



MDS belongs to the so-called force-based methods. MDS actually represents a large set of dimension reduction algorithms that are commonly used in statistical analysis and information visualization. The main goal is to try to preserve the properties of N-dimensional data when projecting into another dimension (e.g., the relationships existing between the data in the original dimensions should be preserved even after the projection). The projection may introduce certain artifacts that may appear in the resulting visualization and were not included in the original data at all.

A typical MDS algorithm has the following structure:

1. Let us have a dataset with M records and N dimensions. We create an MxM matrix Ds containing the results of measuring the similarity between individual pairs of input data. This

measurement can be performed in various ways, for example using Euclidean distance metrics.

2. Suppose we want to project the input data into K dimensions (for display purposes, K is usually between 1 and 3). We construct a matrix L with dimensions MxK, which contains the location of the projected points. These M locations may be chosen at random or a technique such as principal component analysis may be used.

3. Calculate the matrix Ls with dimensions MxM, which contains the similarity between all pairs of points from L.

4. Calculate the value of the so-called stress - S, which is determined by measuring the differences between Ds and Ls.

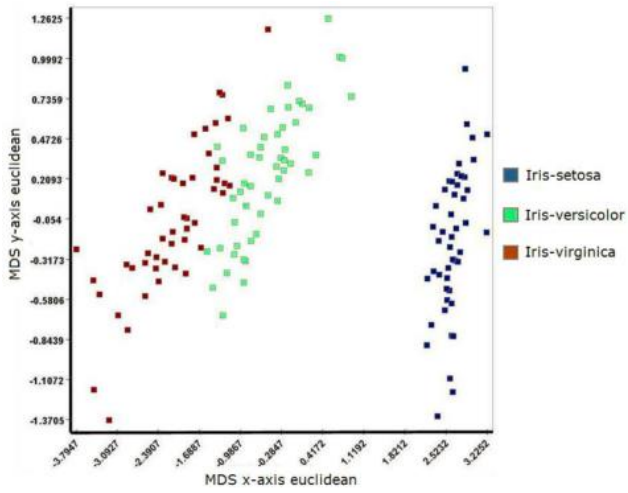
5. If S is small enough or has not changed significantly in the last few iterations, the algorithm ends.

6. Otherwise, we shift the positions of the points in L in a direction that reduces their individual stress values. This can be, for example, a weighted sum of the displacements based on a point comparison with all other points or only with the nearest neighbors.

7. Return to step 3.

Obviously, there are several variants of this algorithm. Their difference lies mainly in the way of calculating the similarity and value of stress, in the different definition of initial and final conditions and in the different strategy of updating the position of points. As with other optimization algorithms, it may happen that we can get stuck in the local minimum, which, however, still has a high value of stress. Common strategies that address this problem occasionally add a random "jump" to a given point position where the goal is to converge to another location.

The figure shows an example of an iris dataset showing the four numerical dimensions where the MDS technique was used for projection.

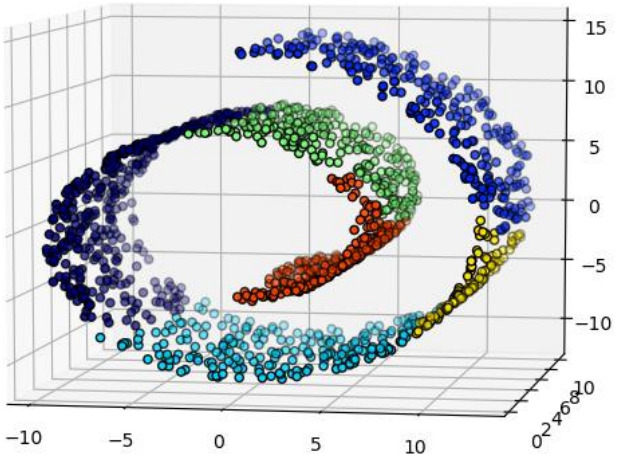


Problems:

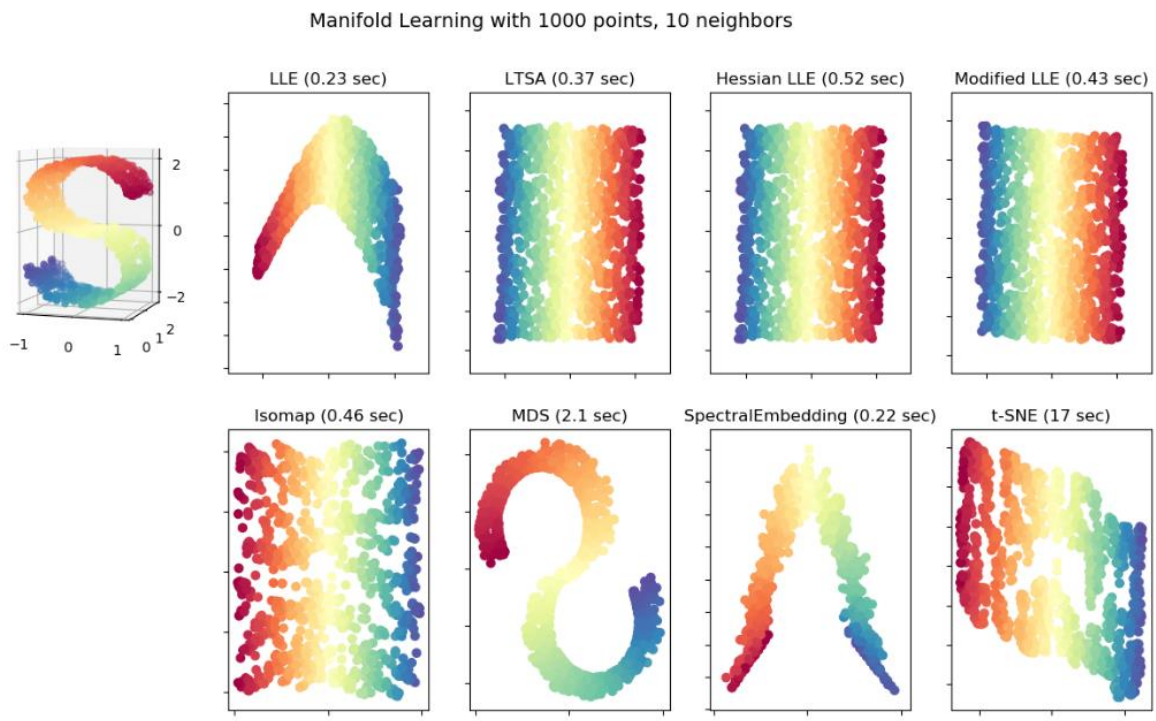
For most of the techniques we present in this section, the results are not unique: small changes in the initial conditions can lead to completely different results. Another problem is that the coordinate system after projection is not completely "meaningful" for the user - due to the dimensions of the original data. For example, it is typical to map a data point to a position at the top of the display when one algorithm is running, and to map a data point to a position at the bottom of the display when running another algorithm that solves the same problem. Therefore, the relative position of the individual points is important, not absolute.

Non-linear projections

In these techniques, the low-dimensional surface is embedded non-linearly in the high-dimensional space. The outcomes of these methods can possess different shapes (one of the typical is the spiraling band or the Swiss roll – see figure). The goal of these methods is to preserve the information about the neighborhood between points in the high-dimensional space. They are locally linear and are based on the pairwise distances. However, on larger distances, the non-linearity causes the problem that we cannot conclude that the distant points in the reduced space are also distant in the original space.



In the following, you can see the outcomes of several non-linear DR techniques, such as isomaps or locally-linear embedding (LLE).



Hybrid approaches

Dimensionality reduction is often unwanted because the domain knowledge of the user is often required to understand which dimension combinations are meaningful. Therefore, it is beneficial to combine the user-defined feature selection (based on visual analysis approaches using different quality metrics) with subsequent feature extraction, that is performed only on the selected dimensions. For the visual representation, we can utilize the above-mentioned techniques.