Parallel Coordinates Technique

Data Wrangling

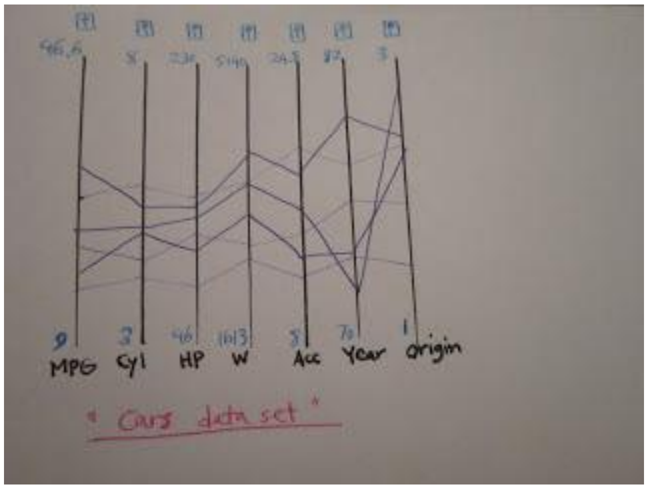
At first, I decided to convert data set to an appropriate format so that I can use "FloatTable.pde" so I did what follows:
I wrote a short function in matlab to separate columns of "cars.okc" file,I stored each column in a separate file (So I got 7 files, each for values of one attribute of cars data set). Meanwhile I opened an excel file and typed each attribute of cars data set in header of one column.Then I copied and pasted data of each file to the corresponding column of excel file
My code in matlab for separating the columns is this :

```
function []= f()
cars = load(['cars.okc']);
  for i=1:7
     d = cars(:,i);
     saveFile =['save -ascii '  'cars_' num2str(i) '.okc d'];
     eval(saveFile);
  end
end
```

Now, I can use "FloatTable.pde" file reader to read cars data set.
But then I realized it is easier to write this homework in eclipse environment because it equipped me with auto-completion, auto-indent, and some other features, so I created a new project namely "ParallelCoordinates" in eclipse and added core.js of processing library to its build path. Now we can implement processing code in eclipse environment.
For handling data set, I wrote a class namely "dataReader.java" which is in charge of reading data set file from a specific give address, parsing it, storing it into a 2 dimensional array and tasks such these. It is also able of taking kmeans parameter and reading cluster files from a given address.
For working with "cars.okc" data set, I removed first line of this file, wrote features names _which were written in separated lines_ in one line, so I achieved a neat file which has features names in first line and one record in each other lines, now it is ready to be processed. For "cameras" data set, I copied the data into a text file and remove some lines which had null values.

Basic Visualization

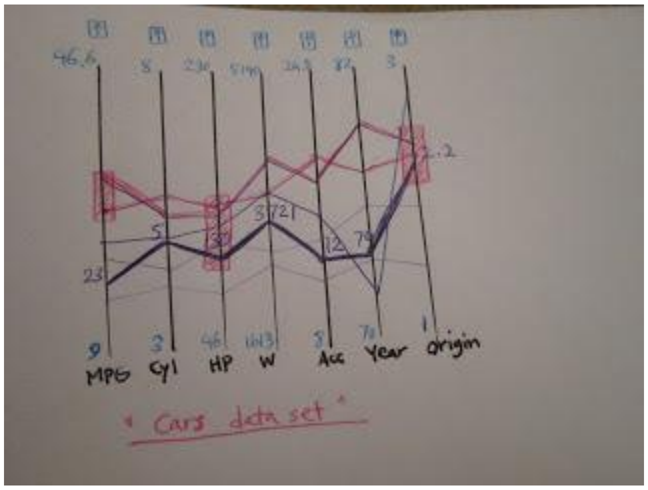
This is my first version prototype paper:



Small arrows above each axis, shows the fact that axis is in ascending order or descending order, in this picture, all arrows are up which shows all axes are in ascending order. If we click on arrow, its direction will reverse and the corresponding axis will be ordered in descending order. Below each axis, name of corresponding feature is written, also minimum and maximum values of each axis (range of each axis) are written. at bottom of the picture, you see the title of this chart "cars data set". Note that transparency is used in this picture to distinguish overlapping lines from each other (some lines are lighter and some are darker).

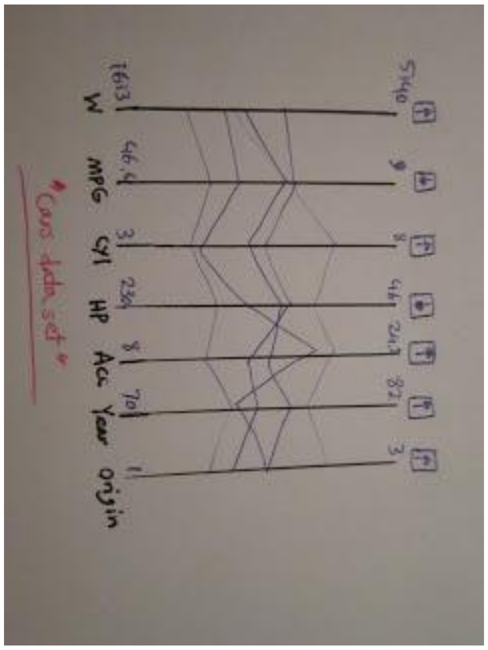
Interactivity

These are my paper prototype for interaction part, each one demonstrates two interaction methods:



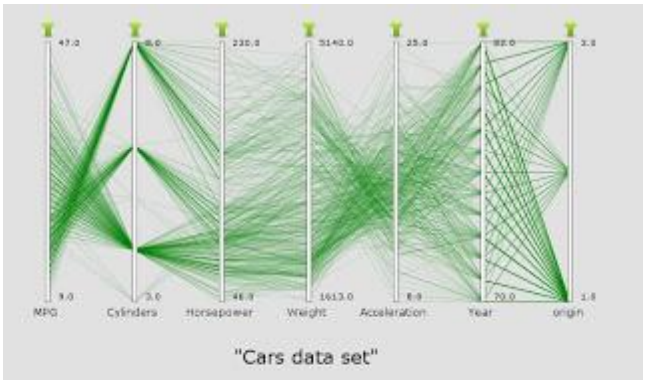
This picture offers 2 ways of interaction with data which are as follows:

1) highlighting and showing details: by moving cursor over a line, the line will be bold and its values across different access will be shown. In this picture one line is bold.
2) filtering: by selecting some parts of different axes, color of lines which pass through these parts will be changed, in this way these lines will be distinguished from others. In this picture, filtered lines are shown in pink color.

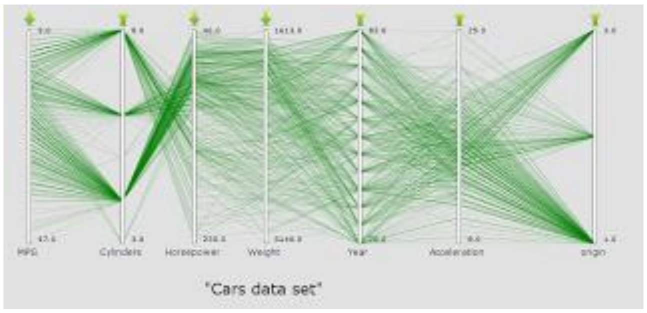


This picture offers 2 other ways of interaction with data:
1) reorder axes: by dragging each axis, we can change its position, put 2 different axes near each other and see their relationship. As you see in this picture, W axis has moved to the first position.
2) invert axes: by clicking on arrow symbol above each axis, we can change the ordering of axis from ascending to descending or vice versa. In this picture, MPG and HP axes have been inverted.

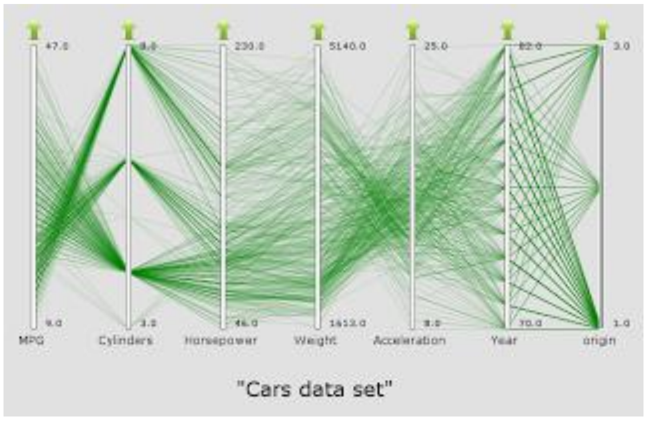
I implemented the code related to this part, for that I create a java class namely "axis.java", each feature of the data set is an object of this class. Also I created a "Main.java" class which is the coordinator and handles tasks related to axes.
The picture below is the output of my code which shows all lines from "cars data set", notice that lines are shown with different transparency so that overlapping lines are visible.



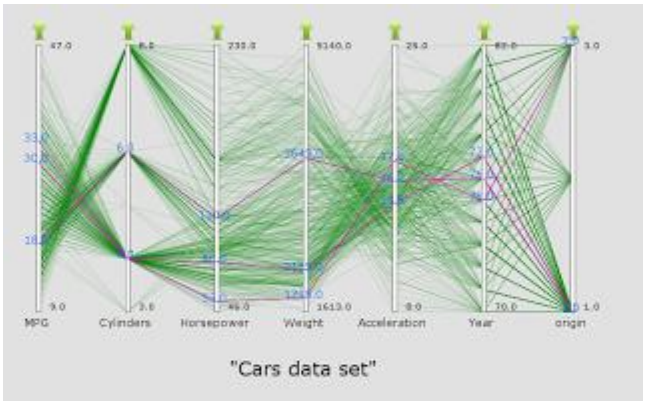
By pressing arrow image above axes, we can invert axes. Also we can drag axes and changed their position like below :



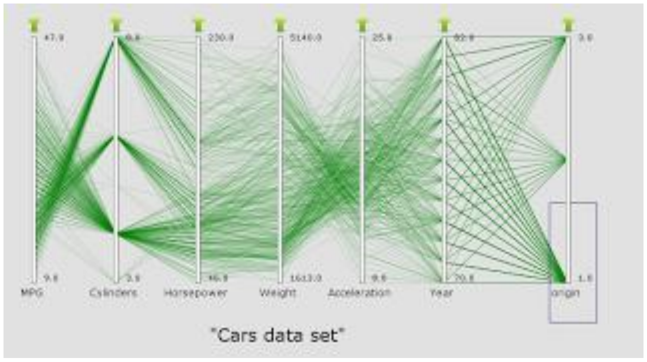
In this picture, "MPG" , "HorsePower" and "Weight" are inverted. position of "Year" and "Acceleration" axes are changed. Also notice that distances between axes are changed so that some lines which overlapped could be seen easier.
Also when user move the mouse cursor on one axis, the color of axis changes to gray and when user press mouse button on the axis, its color changes to white as an indicator that some operation might be doable. In picture below, mouse is over last axis ("Origin").



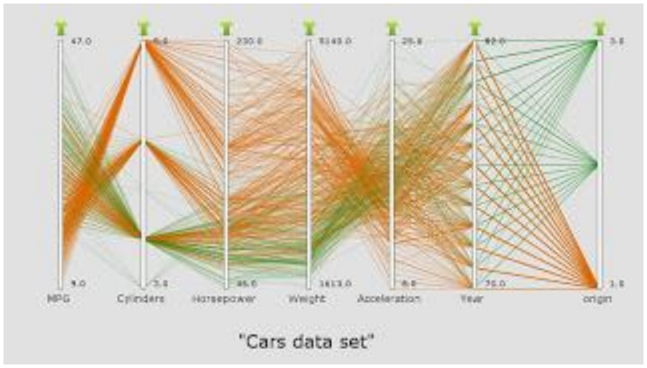
When user goes over one line or the intersection of some lines, their color will be changed to red and values of them across all axes will be shown in blue :



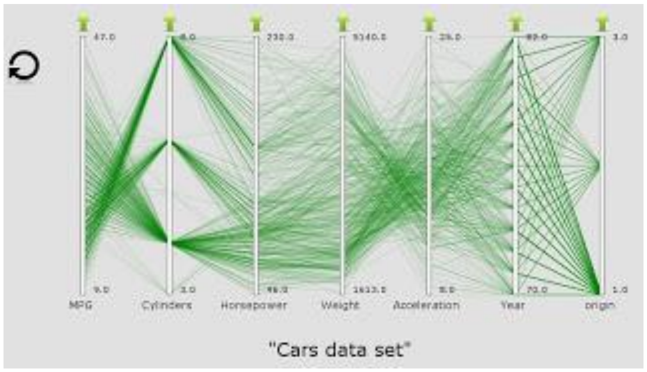
Also, user might select some portion of axes, selecting box will be shown on the chart as below:



and the result of the filtering is as follows: (filtered lines are shown in orange color)



for disabling filtering, user should just click on the screen (some where except axes), in this way only filtering will be removed but all reordering and inverting that has done before will remain. If user wants to return back to the first state of the diagram (primary order of axes and etc.) he/she should press the restart button at the left side of diagram:



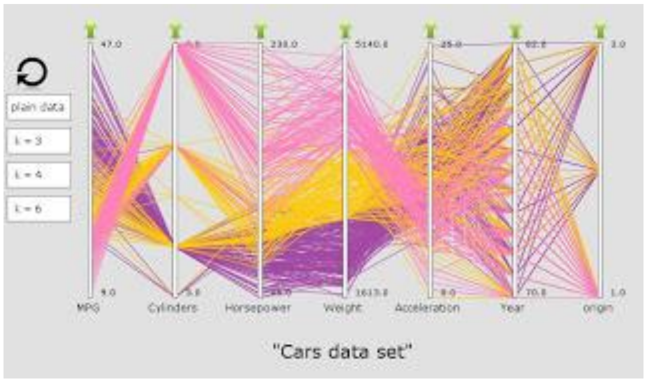
Clustering

For this part, I wrote a code in Matlab which uses kmeans function of matlab and clusters data, my code is as follows:

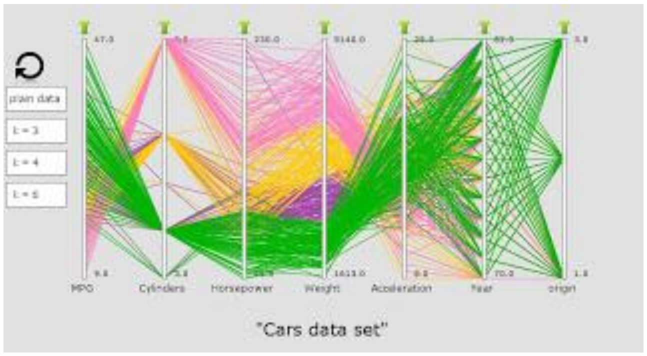
```
function [] = kmeansClustering()
    dataMatrix = load(['cars' '.okc']);
    kmeansp = 6;
    dim = 7;
    idx = kmeans(dataMatrix ,kmeansp);

    file = zeros(kmeansp,size(dataMatrix,1), dim);
    dataIds = zeros(kmeansp,size(dataMatrix,1));
    counter = ones(kmeansp,1);

    trainIdsFile = 1:392;

    for i=1:size(dataMatrix,1)
        j = idx(i);  %j = id of cluster
        a = counter(j);   %a = row number
        dataIds(j,a) = trainIdsFile(i);
        file(j,a,:) = dataMatrix(i,:);
        counter(j) = counter(j) + 1;
    end


    for i=1:kmeansp
        for j=1:counter(i)-1
            realIndex(j) = dataIds(i,j);
        end
        saveF=['save -ascii ' 'ids_cluster' '_' num2str(i)
'.txt realIndex'];
        eval(saveF);
    end

    for i=1:kmeansp
        temp = zeros(counter(i)-1,dim);
        for j=1:counter(i)-1
            for k=1:dim
```

```
                temp(j,k) = file(i,j,k);
            end
        end
        saveFile =['save -ascii ' 'data_cluster' '_' num2str(i)
'.txt temp'];
        eval(saveFile);
    end
end
```

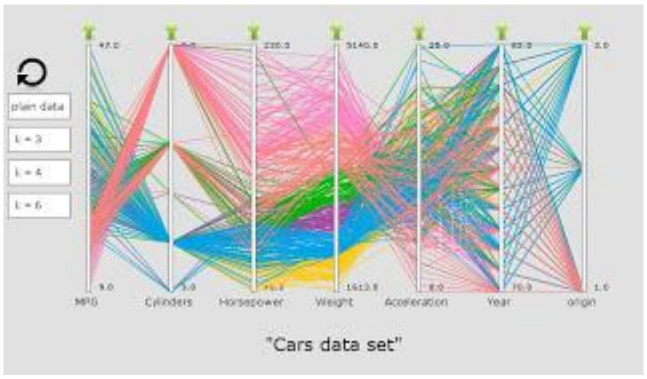
This code produces kmeansp cluster files, I gave these files to dataReader.java class, this class reads all clusters and store them in arrays, the Main.java class will uses these arrays to show clusters visually. I tested data set with several kmeans parameter, the result for k=3, k=4 and k=6 are shown in the screen so that user can change between them, each cluster is shown via a separate color. the result for k=3 is as follows:



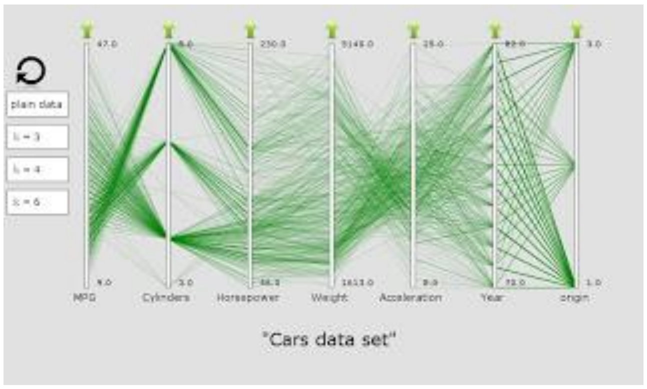
By clicking on k=4 button, the result for 4 clusters will appear :



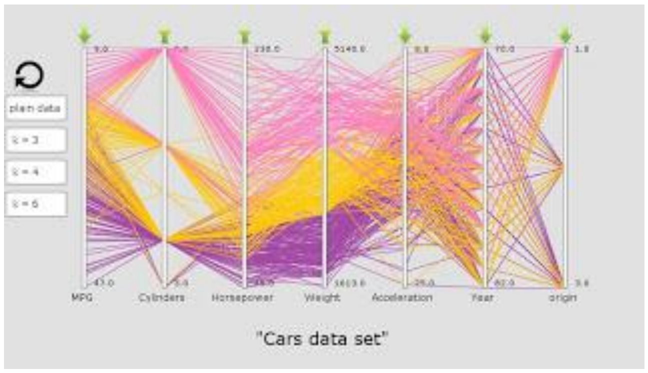
By clicking on k=6 button, the result for 6 clusters will be shown which is as follows:



If user clicks on plain data button, all clusters will disappear and plain data will be shown:



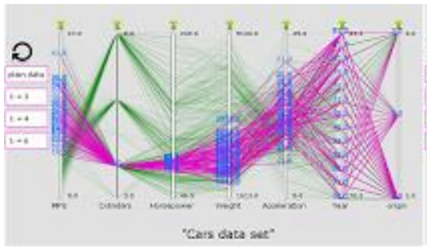
Among all experiment with the number of clusters which I did, I think k=3 works best and creates meaningful clusters, if we invert some axes, we reach to a better representation of data when we have 3 clusters:

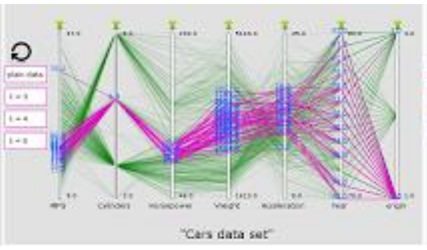


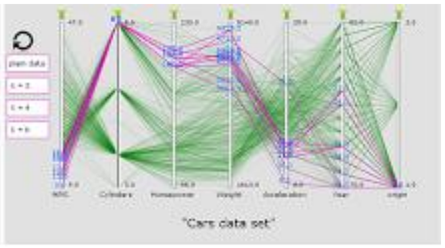
As you see in this picture, clusters are rather well separated.

Data Exploration
For cars data set:

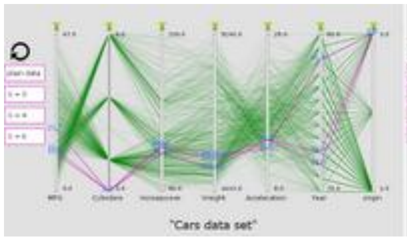
Each polyline represents a different car, polylines are drawn with a certain amount of transparency (40), in this way, areas covered by many lines, for example when the data are correlated appear darker on the plot.
 I found several interesting facts in this visualization. Groups of lines that run parallel indicate similar data points, lines widely spread apart along an axis show a large variation of of that data attribute. For example we can see that most cars have 4, 6 or 8 cylinders (look at the pic below, it shows that 3 groups of lines are existing from "Cylinder" axis)

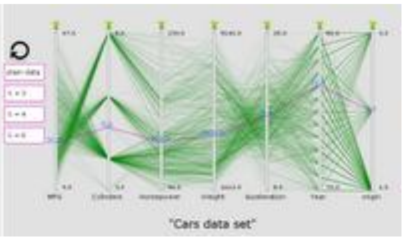




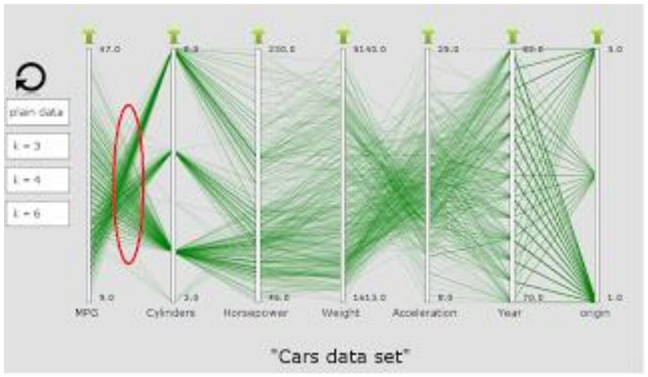


Also we can see that there are very few cars with 3 or 5 cylinders:

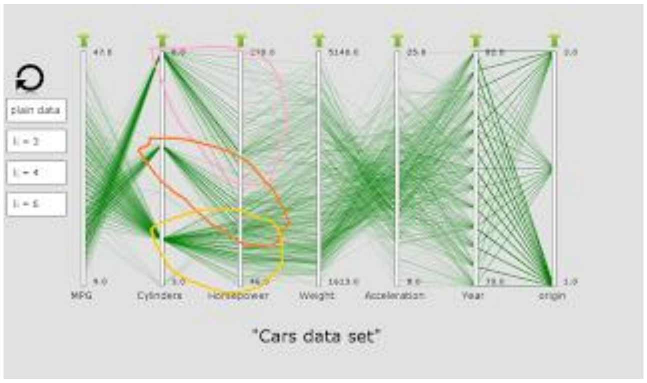




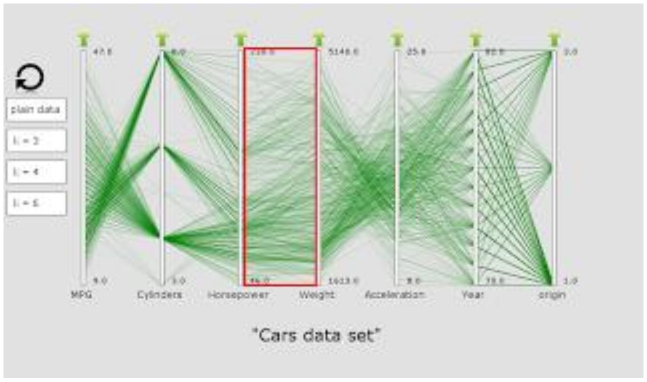
We can easily see the inverse correlation between MPG and number of cylinders, those cars which have higher MPG, have lower number of cylinders and those with lower MPG have larger number of cylinders, this inverse correlation is detectable by seeing intersection of lines between these two axes :



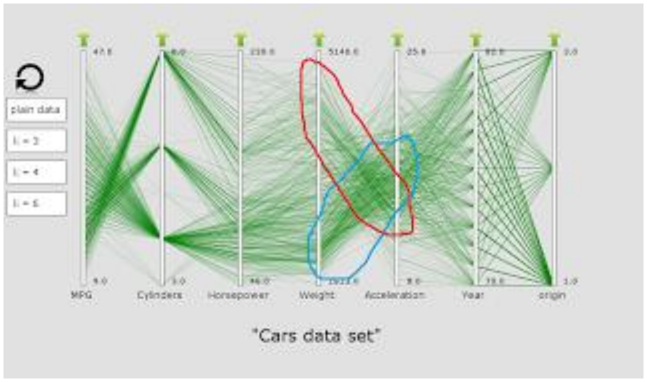
It is observable that there is a direct correlation between number of cylinders and horsepower, those cars who have fewer number of cylinders have less horsepower and those who have more cylinders have more horsepower, although we can also say that fan-out of lines going from cylinder axis to horsepower axis is slightly large, these lines have divided horsepower axis into 3 parts:



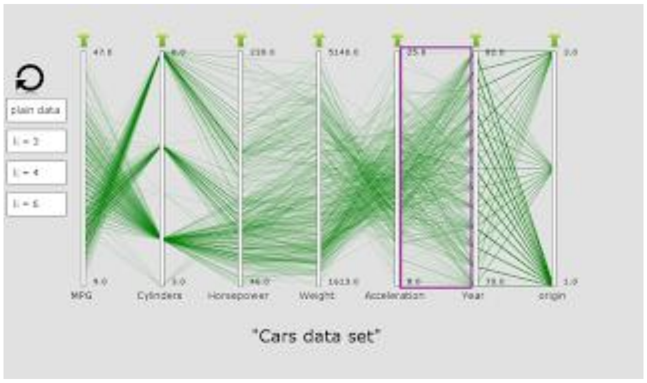
Lines from horsepower to weight run mostly parallel, which shows these two features are strongly proportional, cars with more horsepower have more weight and vice versa.



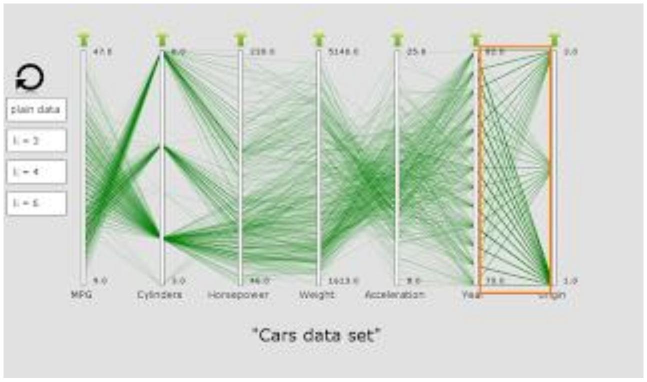
We can see an inverse correlation between weight and acceleration axes, cars which have more weight have less acceleration and vice versa :



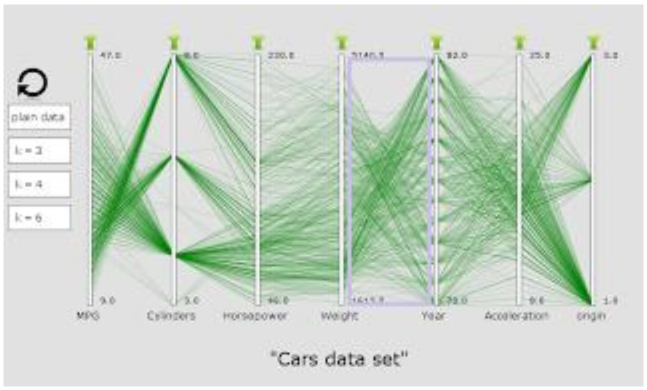
The correlation between year and acceleration is interesting. Almost the similar bundle of lines emerge from year axis and go to acceleration axis, it shows that similar number of cars are produced every year and they have come in all acceleration categories :



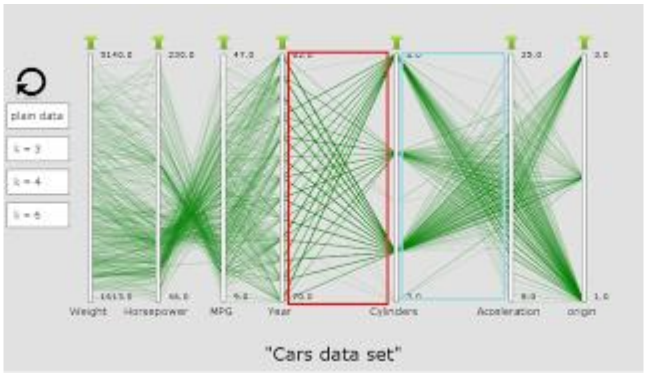
Fan-out of lines going from origin axis to year axis is strong, this indicates that origin does not indicate production year, also production year does not indicate origin, which seems reasonable.



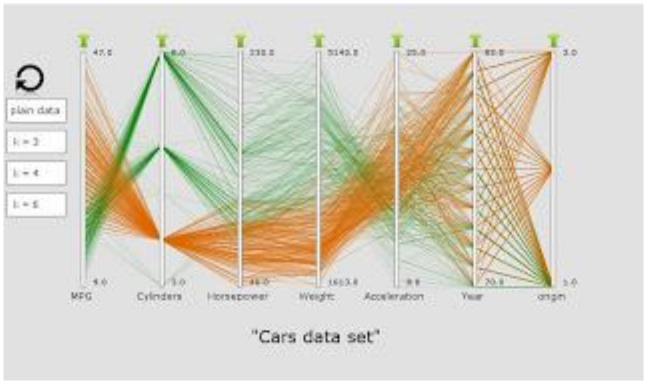
By moving year axis and putting it near weight axis, we see that there is an inverse correlation between them , cars produced in later years have less weight while those produced in early years are heavy:



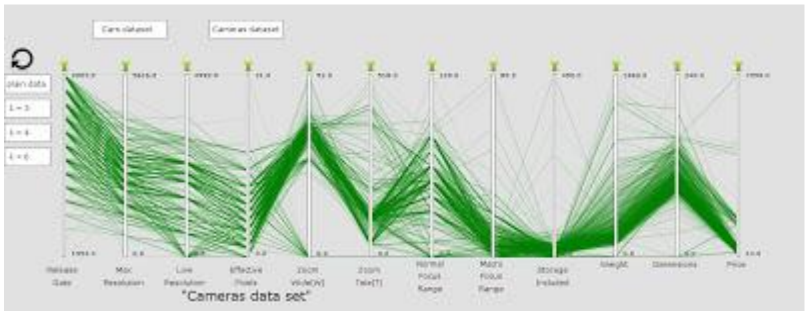
In the picture below, we can see two facts :
1) the relation between accelerator and cylinders, which is an inverse relation and suggests that generally cars with more cylinders have less acceleration and cars with less cylinders have more acceleration
2) the relation between cylinder and year, which implies that in each year, cars with different number of cylinders have been produced.



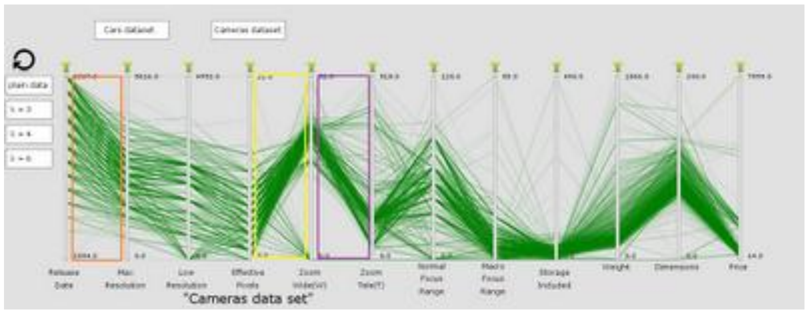
Furthermore, by brushing and filtering axes we can see more interesting facts, for example we see that majority of cars have four cylinders, they have high MPG, low horsepower and low weight but high acceleration and they are more produced in later years :



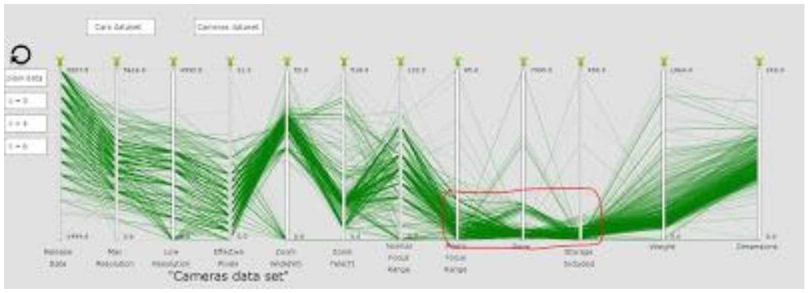
For Cameras data set:
You can switch between camera and car data set by clicking on two buttons above the diagram:



The picture above shows cameras data set. Here we can see several interesting facts. There is a direct correlation between release date and max resolution of camera, this relation shows in later years, max resolution of cameras has increased. Also we can see that there is an inverse relation between zoom wide and zoom tele and also between zoom wide and effective pixels :



Also we can see that most cameras have low price, low storage and low macro focus range:



About Parallel Coordinate as a visualization
Parallel coordinate have some limitations that prevent it from being widely used for all type of data sets. For large and complex data sets which have large number of

attributes, parallel coordinate does not work because it will produces cluttered diagrams which are difficult to understand and interpreting. Also ordering of axes plays a great role here as correlations are understandable only between direct neighbor axes. So when the number of features increases it is rather impractical to find out all meaningful correlations because it takes a huge amount of time and interaction to brushing axes, reordering them and etc. I guess most users might be unfamiliar with parallel coordinate and how it works, so another flaw of it is that it needs a little training to be understood by users.
On the other hand, parallel coordinate offers several great advantages. first of all it allows us to show multidimensional data via a simple structure in 2-D space. The structure of parallel coordinates allows many interaction and visual enhancements be build on it and hence many investigation tasks can be processed. It can be integrated with other techniques such as scatter plots and statistical analysis tools to facilitate combination of different exploration tasks. Also there are some techniques including hierarchical clustering that help reduce clutter when visualizing large data sets with parallel coordinate.

I myself, found parallel coordinate easy to use, and I used a few interactivity to find correlations, maybe that's because the data sets I used ("cars" and "cameras") do not have many features.
The limits of parallel coordinates for data other than quantitative data is that they should some how be converted into quantitative ones, for example for categorical (or qualitative) data we can do the conversion and assign a number to each category but then the correlation between axis representing categorical feature and other axes are meaningless, for example assume that for one data set there are two features "price" and "color", color is categorical while price is quantitative, if we assign a number to each color and then draw data points as lines, then we can not interpret any meaningful relations. It is better to not represent categorical features as axis in parallel coordinate, for showing the value of a categorical feature, you can add mouseOver event such that when mouse goes over a line, the value of categorical features is shown.

You can download the java code from here, for running it you just need to create a new java project and add core.jar file to its build path and correct the absolute address of the files (in Main class) based on the place you save them. Also you can download processing version code from here.