This Document is just a first version to explain the important functions. An updated version will follow soon. For questions please write to sascha.henzgen@upb.de.

August the 29th 2014, Paderborn.

FisVis is an in Java implemented visualization tool. For the GUI elements Java Swing has been used. For some calculations we also used the *commons math library* from the Apache Commons Project (`http://commons.apache.org/`).

1 How to run FISVis

Their are two ways to run FISVis. The first is to connect FISVis to an algorithm and visualize the models as soon the come out of the algorithm $\mathcal{A}$. The second way is to run a stored FISVis session.

1.1 Connecting FISVis to your algorithm $\mathcal{A}$

When FISVis is used to visualize the models produced in an algorithm, the user has to call two functions from the `class FISVis`, namely:

```
public void startSession(double[][] mean_A
                                    , double[][] width_A
                                    , double[][] out_A
                                    , int[] rule_to_ID
                                    , int[][] ancestors
                                    , int max_ID
                                    , double[][] attr_range
                                    , double[] out_range
                                    , double[][] data
                                    , int numDataAll
                                    , String url)
```

and

```
public void updateSession(double[][] mean_A
                                    , double[][] width_A
                                    , double[][] out_A
                                    , int[] rule_to_ID
                                    , int[][] ancestors
                                    , int max_ID
                                    , double[][] attr_range
                                    , double[] out_range
                                    , double[][] data
                                    , int numDataAll)
```

The method `startSession(...)` is invoked at the beginning. It initialize the visualization with the rule system $\mathfrak{R}^{(0)}$. For every new rule system $\mathfrak{R}^{(T)}$ with $T > 0$, which should be given to FISVis, the method `updateSession(...)` has to be called.

Discription of parameters:

- `mean_A[i][j]`: $c_{i,j}$ - Center of the j-th fuzzy set $\mu_{i,j}$ fo the i-th rule $R_{id_T(i)}$.
- `width_A[i][j]`: $\sigma_{i,j}$ - Width of the j-th fuzzy set $\mu_{i,j}$ fo the i-th rule $R_{id_T(i)}$.
- `out_A[i][k]`: $\omega_{i,j}$ is the i-th coefficient of the linear function l_j belonging to the conclusion of the i-th rule $R_{id_T(i)}$.
- `rule_to_ID[i]`: $id_t(i)$ - Contains the ID of the i-th rule.
- `ancestors[i][d]`: ID of the d-th parent rule of the i-th rule, with $d = \{0, 1\}$.

- `max_ID`: The highest assigned ID in all generations $t \leq T$.
- `attr_range[p][d]`: Minimal (maximal) attribute value for $d = 0$ ($d = 1$).
- `out_range[d]`: Minimal (maximal) coefficient for $d = 0$ $d = 1$.
- `data`: $\mathcal{S}$ the data set which let the algorithm $\mathcal{A}$ produce an update $\mathcal{A}(\mathcal{S}, \mathfrak{R}^{(T-1)}) = \mathfrak{R}^T$
- `numDataAll`: The overall amount of data points presented to the algorithm.
- `url`: A string containing the URL to the directory where the session and data should be stored.

To store the data is needed for the dynamic parallel coordinates, where also the data can be visualized. But not the whole data, only the data responsible for the currently watched generation. Since a stream session can be very long the data is not hold in the main memory but stored on the hard drive. As the data set, belonging to one generation, is normally not so large, reading from the hard drive should not be a problem.

1.2 run a stored session

2 Example: Run a stored session

3 Example: Running FisVis from Java

4 Example: Running FisVis from Matlab

```
%...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% start of visualization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

javaaddpath('C:\Dokumente und Einstellungen\Sascha\workspace\Master\bin');
javaaddpath('C:\Programme\Java\commons-math3-3.0-bin\commons-math3-3.0\...
                        commons-math3-3.0.jar');

java_gui_adapter = adapter.FlexFisAdapter;

ruleInfo = [];

ruleInfo = visualizeFIS(fismat, ruleInfo, java_gui_adapter,...
                        false, data(1:startData,1:numOfAttributes),startData);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% first update of visualization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ruleInfo = visualizeFIS(fismat, ruleInfo, java_gui_adapter,...
                        true, data(startData+1:startData+1+incrementalStep,...
                        1:numOfAttributes),startData+1+incrementalStep);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%...


for j=startData+1+incrementalStep+testStep:jump:size(data)

%...
```

```
%#########################################################################

ruleInfo = visualizeFIS(fismat, ruleInfo, java_gui_adapter,...
                        true, data(j+1:j+incrementalStep,1:numOfAttributes),...
                        j+incrementalStep);

%#########################################################################

%...
```