# Network analysis with Cytoscape

presented by Kai, Stefan & Stephan

in lecture on systems bioinformatics

02/08/2012

# Outline

# Outline

# Introduction I

. . . why should we care about glycolysis and gluconeogeneses

# Introduction II

. . . basic setup of our two networks ... how we modelled them and what did not work (CellDesigner) ...

# Outline

# Plugin for calculation of Katz's Index

## Definition (Katz's Index)

...

We considered two cases: $\alpha$ small and $\alpha = \frac{1}{\lambda_{max}}$.

# Plugin for calculation of Katz's Index - $\alpha$ small

**Listing 1: Necessary import statements**

```
1  package katz.plugin;
2
3  import java.util.List;
4  import java.awt.event.ActionEvent;
5  import javax.swing.JOptionPane;
6  import java.text.DecimalFormat;
7
8  import cytoscape.plugin.CytoscapePlugin;
9  import cytoscape.util.CytoscapeAction;
10 import cytoscape.Cytoscape;
11 import cytoscape.CyNetwork;
12 import cytoscape.CyEdge;
13 import cytoscape.view.CyNetworkView;
14
15 import Jama.Matrix;
```

# Plugin for calculation of Katz's Index - $\alpha$ small

**Listing 2: Basic class definition**

```
16 public class KatzPlugin extends CytoscapePlugin {
17
18     public KatzPlugin() {
19         KatzPlugin.MolbiPluginAction action = new KatzPlugin.MolbiPluginAction();
20         action.setPreferredMenu("Plugins.Molbi");
21         Cytoscape.getDesktop().getCyMenus().addAction(action);
22     }
23
24     public String describe() {
25         return "Plugin to calculate the topological Katz Index for networks";
26     }
27
28     public class MolbiPluginAction extends CytoscapeAction {
29
30         public MolbiPluginAction() {
31             super("Katz's Index (alpha = 0.1)");
32         }
33
34         @Override
35         public void actionPerformed(ActionEvent ae) {
36             run();
37         }
```

# Plugin for calculation of Katz's Index - $\alpha$ small

**Listing 3: Run method (I)**

```
38          private void run() {
39
40              CyNetwork network = Cytoscape.getCurrentNetwork();
41              CyNetworkView view = Cytoscape.getCurrentNetworkView();
42              int N = network.getNodeCount() + 1;
43
44              if (N == 1) {
45                  JOptionPane.showMessageDialog(view.getComponent(), "No network/view
                        loaded.");
46                  return;
47              }
48
49              double[][] A = new double[N][N];
50              for (double[] row : A) Arrays.fill(row, 0.0);
51
52              for (CyEdge edge : (List<CyEdge>) network.edgesList()) {
53                  int i = Math.abs(edge.getSource().getRootGraphIndex());
54                  int j = Math.abs(edge.getTarget().getRootGraphIndex());
55                  A[i][j] = 1;
56              }
```

# Plugin for calculation of Katz's Index - $\alpha$ small

**Listing 4: Run method (II)**

```
57                  Matrix M = new Matrix(A);
58                  Matrix I = Matrix.identity(N, N);
59                  Matrix IVec = new Matrix(N, 1, 1.0);
60
61                  double alpha = 0.1; // emulates degree centrality
62                  double[][] values = ((I.minus(M.transpose().times(alpha))).inverse()).
                        minus(I).times(IVec).getArrayCopy();
63
64                  StringBuilder sb = new StringBuilder();
65                  DecimalFormat df = new DecimalFormat("#0.000");
66
67                  for (int i = 1; i < N; i++) {
68                      if (i % 10 == 0) sb.append("\n");
69                      sb.append("C(").append(i).append("): ").append(df.format(values[i
                            ][0])).append("    ");
70                  }
71
72                  JOptionPane.showMessageDialog(view.getComponent(), "All done.\n" + sb);
73                  view.redrawGraph(false, true);
74              }
75          }
76      }
```

# Plugin for calculation of Katz's Index - $\alpha = \frac{1}{\lambda_{max}}$

## Listing 5: Run method (II)

```
57                Matrix M = new Matrix(A);
58                Matrix I = Matrix.identity(N, N);
59                Matrix IVec = new Matrix(N, 1, 1.0);
60
61                double[] eigs = M.eig().getRealEigenvalues();
62                double alpha;
63                Arrays.sort(eigs);
64
65                if (eigs[eigs.length - 1] == 0) alpha = 1.0;
66                else alpha = 1.0 / eigs[eigs.length - 1];
67
68                double[][] values = ((I.minus(M.transpose().times(alpha))).inverse()).
                       minus(I).times(IVec).getArrayCopy();
69                StringBuilder sb = new StringBuilder();
70                DecimalFormat df = new DecimalFormat("#0.000");
71
72                for (int i = 1; i < N; i++) {
73                    if (i % 10 == 0) sb.append("\n");
74                    sb.append("C(").append(i).append("): ").append(df.format(values[i
                           ][0])).append("    ");
75                }
76
77                JOptionPane.showMessageDialog(view.getComponent(), "All done.\n" + sb);
78                view.redrawGraph(false, true);
79            }
80        }
81    }
```

# Plugin for calculation of Randic's Index

## Definition (Randics's Index)

...

# Plugin for calculation of Randic's Index

**Listing 6: Necessary import statements**

```
1  package katz.plugin;
2
3  import java.util.List;
4  import java.awt.event.ActionEvent;
5  import javax.swing.JOptionPane;
6  import java.text.DecimalFormat;
7
8  import cytoscape.plugin.CytoscapePlugin;
9  import cytoscape.util.CytoscapeAction;
10 import cytoscape.Cytoscape;
11 import cytoscape.CyNetwork;
12 import cytoscape.CyEdge;
13 import cytoscape.view.CyNetworkView;
14
15 import Jama.Matrix;
```

# Plugin for calculation of Randic's Index

**Listing 7: Basic class definition**

```
16 public class RandicPlugin extends CytoscapePlugin {
17
18     public RandicPlugin() {
19         RandicPlugin.MolbiPluginAction action = new RandicPlugin.MolbiPluginAction();
20         action.setPreferredMenu("Plugins.Molbi");
21         Cytoscape.getDesktop().getCyMenus().addAction(action);
22     }
23
24     public String describe() {
25         return  "Plugin to calculate the Index of Randic for networks";
26     }
27
28     public class MolbiPluginAction extends CytoscapeAction {
29
30         public MolbiPluginAction() {
31             super("Randic's Index");
32         }
33
34         @Override
35         public void actionPerformed(ActionEvent ae) {
36             run();
37         }
```

# Plugin for calculation of Randic's Index

## Listing 8: Run method

```
38          private void run() {
39              CyNetwork network = Cytoscape.getCurrentNetwork();
40              CyNetworkView view = Cytoscape.getCurrentNetworkView();
41              int N = network.getNodeCount();
42
43              if (N == 0) {
44                  JOptionPane.showMessageDialog(view.getComponent(), "No network/view
                        loaded.");
45                  return;
46              }
47
48              Double IRandic = 0.0;
49              for (CyNode node : (List<CyNode>) network.nodesList())
50                  // otherwise Randic's Index will be +infinity!
51                  if (network.getDegree(node) != 0)
52                    IRandic += Math.pow(network.getDegree(node), -0.5);
53
54              JOptionPane.showMessageDialog(view.getComponent(), "All done.\nRandic
                     Index " + IRandic);
55              view.redrawGraph(false, true);
56          }
57      }
58  }
```

# Outline

# Results - Centralities

- determine the relative importance of a vertex within a graph
- many centraltity concepts were first developed in social network analysis

  –> e.g. how influential a person is within a social network

**Definition (Four measures of centrality)**

1. degree centrality
2. eccentricity centrality
3. closeness centrality
4. betweeness centrality

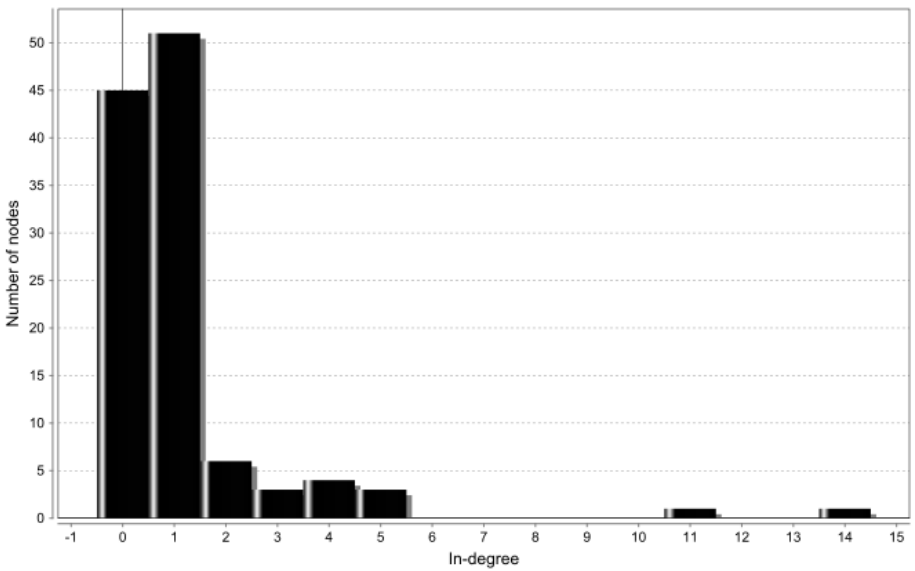
# Results - Degree Centrality

## Definition (Degree Centrality)

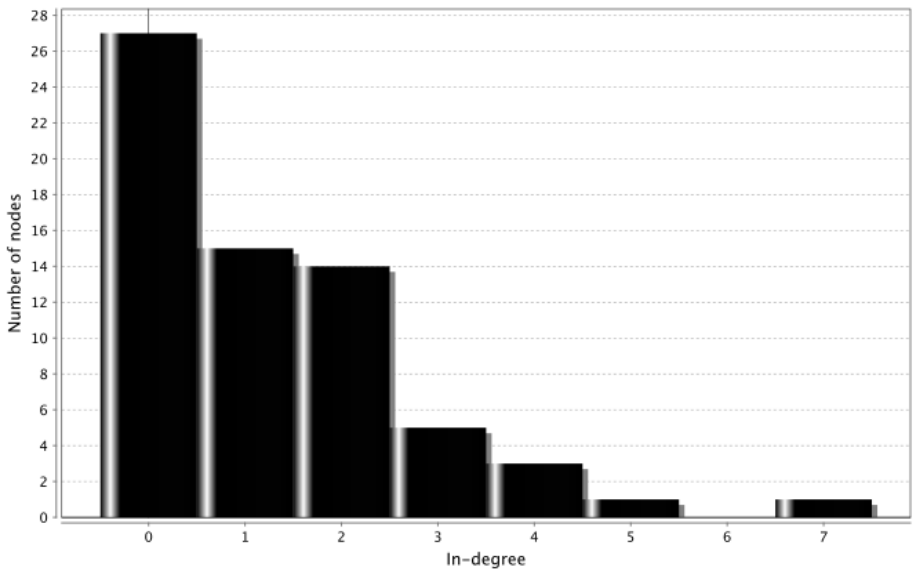
$C_{deg}(v) = \{e | e \in E and v \in e\}$

- counts the number of edges attached to a vertex
- a local centrality measure
  -> only direct neighborhood considered
- directed graphs:
  -> indegree (interpreted as "popularity")
  -> outdegree (interpreted as "gregariousness")

# Results - Degree Centrality



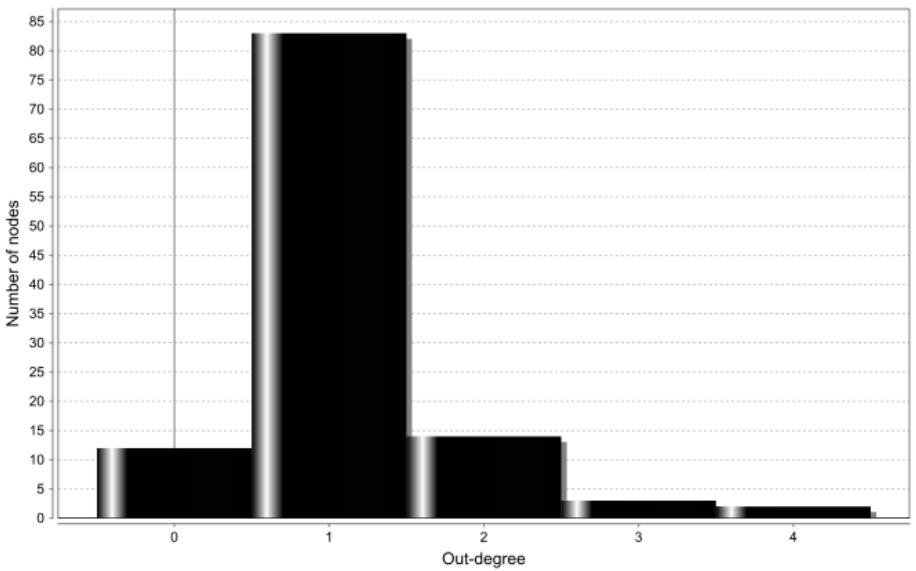

(a) Glyco1



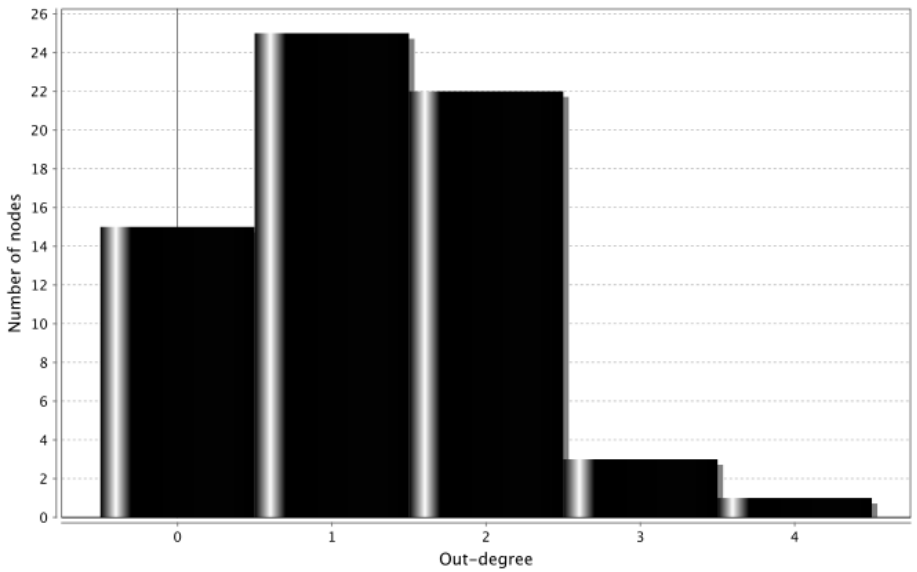

(b) Glyco2

Fig. 1: Indegree Centrality

# Results - Degree Centrality




(a) Glyco1




(b) Glyco2

Fig. 2: Outdegree Centrality

# Results - Eccentricity Centrality

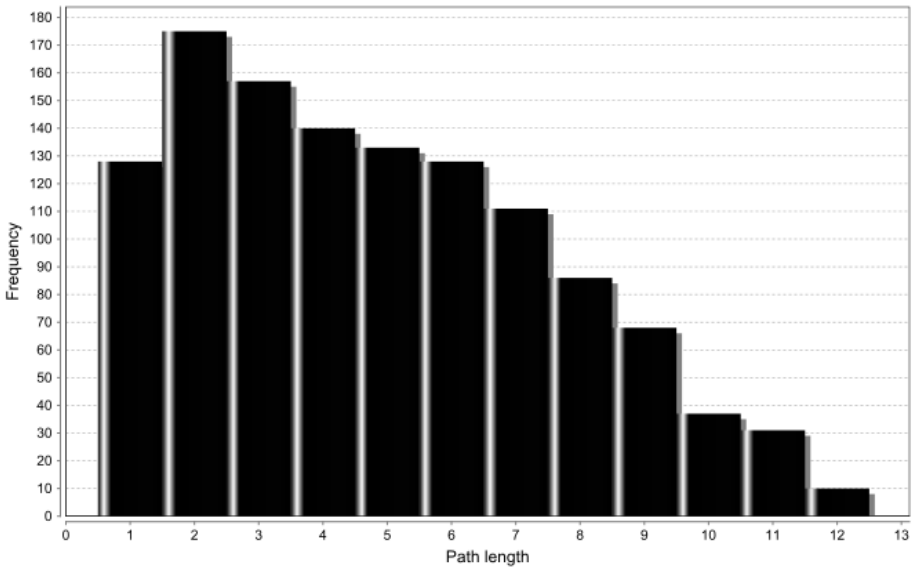
## Definition (Eccentricity Centrality)

$$C_{ecc}(s) = \frac{1}{max\{d_{st} | t \in V\}}$$

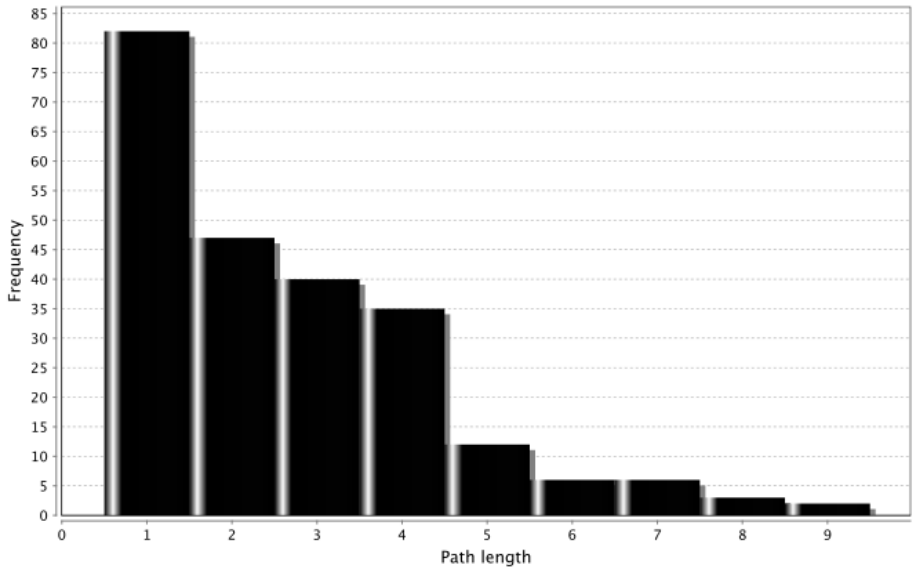
- determine the maximum distance between every two vertices
- central vertices get low values
- centralities require high centrality values

  –> reciprocal is used as centrality value
- only for CONNECTED networks

# Results - Eccentricity




(a) Glyco1




(b) Glyco2

Fig. 3: Eccentricity Centrality

# Results - Closeness Centrality

## Definition (Eccentricity Centrality)

$C_{clo}(s) = \frac{1}{\sum \dots}$

- determine the maximum distance between every two vertices
- central vertices get low values
- centralities require high centrality values

  –> reciprocal is used as centrality value
- only for CONNECTED networks

# Problems

## To other groups

...

# Outline

# Conclusion

## Benefits of cytoscape

...

# Outline

# Resources

## Git repository

git://gitorious.org/cytoscape-plugins/cytoscape-plugins.git