Project Report
Project: Anomaly Detection
Unity Id: ndgandh2

Paper 3 - Event Detection in Time Series of Mobile Communication Graphs

Introduction: This paper contains the implementation of an algorithm that operates on a time-varying network of nodes with edges representing interactions between them and detects **anomalous** points in time at which many nodes **change** their behavior in a way it varies from the normal. In this paper authors have studied the texting behavior of users of an anonymous mobile network in a large city in India. It is a **who-texts-whom** network where nodes represent the users and edges represent the SMS interactions between them. The data consists of six months' of activity of users and is therefore time varying. The report describes the nature and type of graph used for this algorithm followed by the description of the algorithm in brief constraints. Lastly, the analysis of the algorithm and findings from this paper are discussed.

Graph Description: The anomaly detection is performed on the graph data-set of enron, reality mining voices and as-733 data. These data files contain the vertex id pair separated by the new line which is parsed and loaded into graph-like data structures for algorithm implementation. The graphs considered for these algorithm are **Directed and Weighted.**

Algorithm:

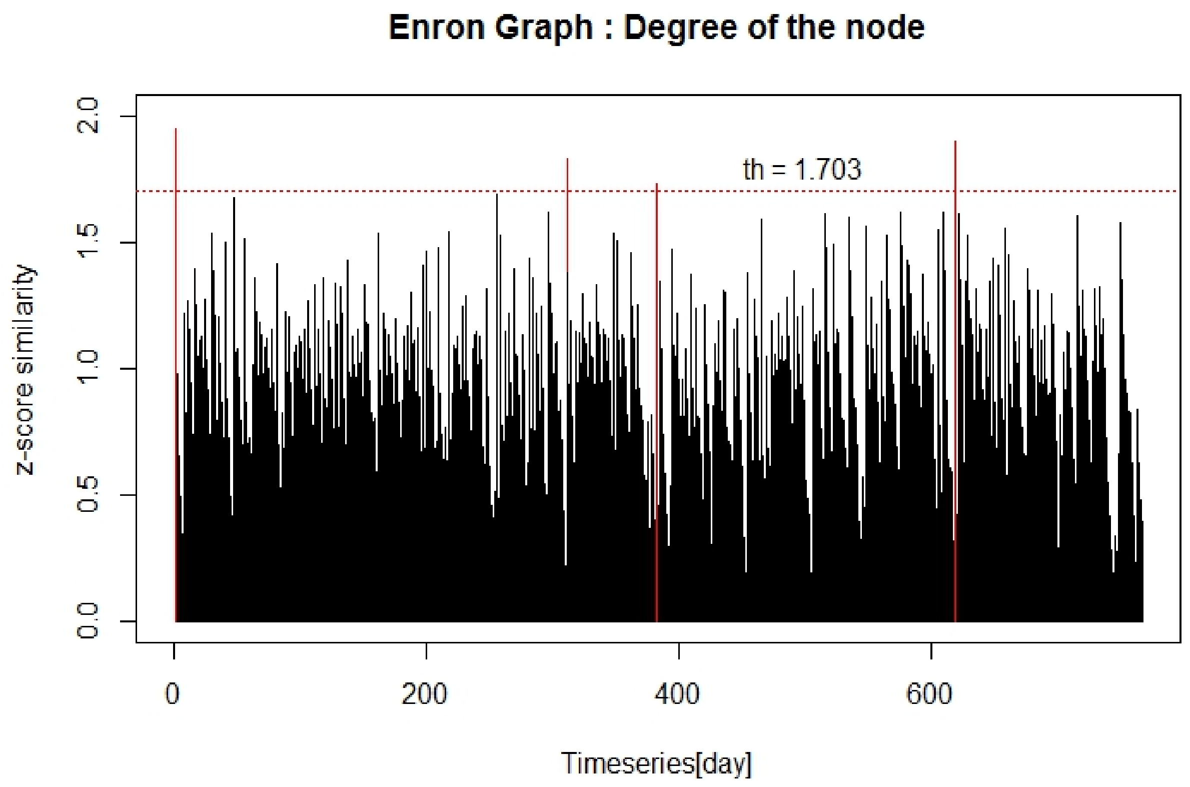
First of all, the graphs for all days of the time series are generated containing N nodes. Then a number of features are extracted for all the nodes to find patterns that the nodes follow. Features extracted are as follows: 1) in-degree, 2) out-degree, 3) out-weight, 5) number of neighbors, 6) reciprocal neighbors, 7) number of triangles 8) in-weight, 9) average out-weight, 10) maximum in weight, 11) maximum out-weight, and 12) maximum weight ratio on reciprocated edges egonet. So the data looks like a 3-D T*N*F tensor where T denotes the number of time ticks, N denotes the number of nodes in our graph and F denotes the number of features extracted for each node. Now for each feature there is a T * N matrix. Next we assume a time window of size W days over the time series of values of all nodes. Then for a pair of nodes , a correlation matrix is computed between the time series vectors over the window of size W. After that for each correlation matrix generated we calculate the principal eigenvector u(t) for current time window. As this algorithm detects contextual anomalies, we calculate the past eigen behavior r(t-1) from the previous X eigenvectors. We take a weighted average of all the previous X eigenvectors using Singular Value Decomposition(SVD) or taking simple weighted average. Now the current eigen behavior vector is compared with past eigen behavior vector by taking a dot product of the two vector. The change metric used is $Z = 1 - u^T r$. Parallely, we calculate the moving range MR and median of the time series based on the z-score and its moving range average $\overline{MR}$ given by the following formula:

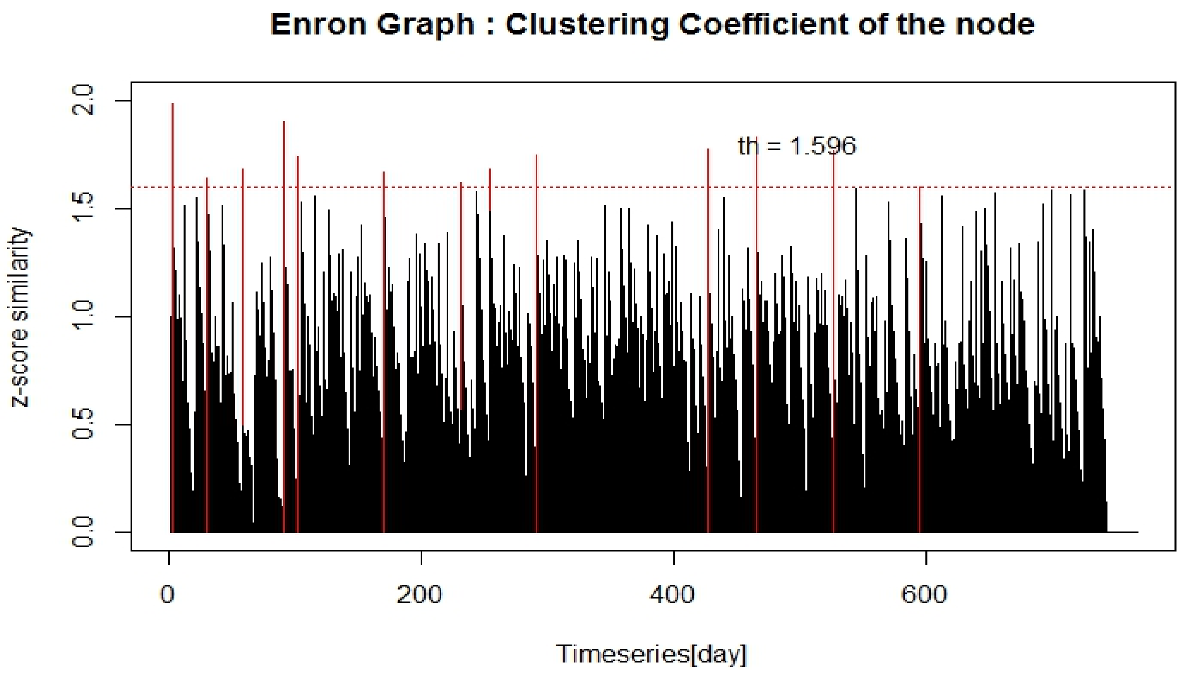
$$\overline{MR} = \sum_{i=2}^{n} MRi \text{ where } MRi = |x_i - x_{i-1}|$$

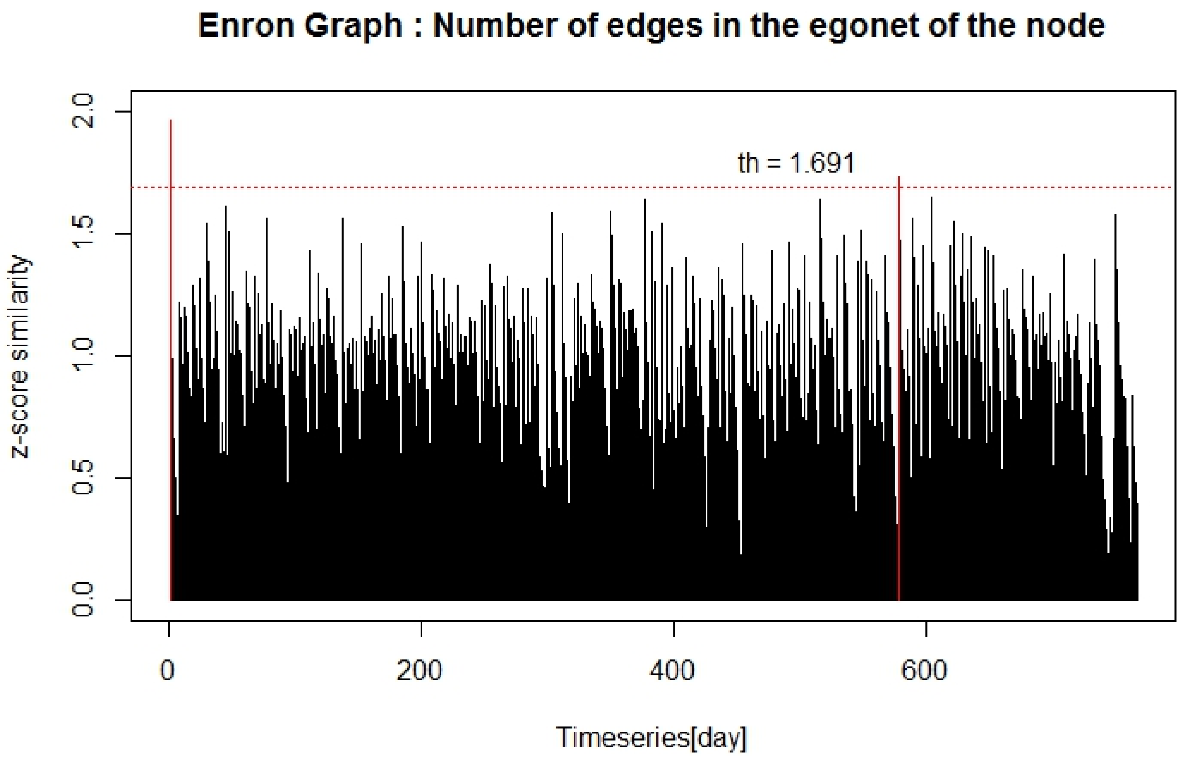
Finally we calculate the upper threshold given by (median + $3\overline{MR}$). If a Z value is above the upper threshold that Z value indicates a change point or an anomaly and is flagged.

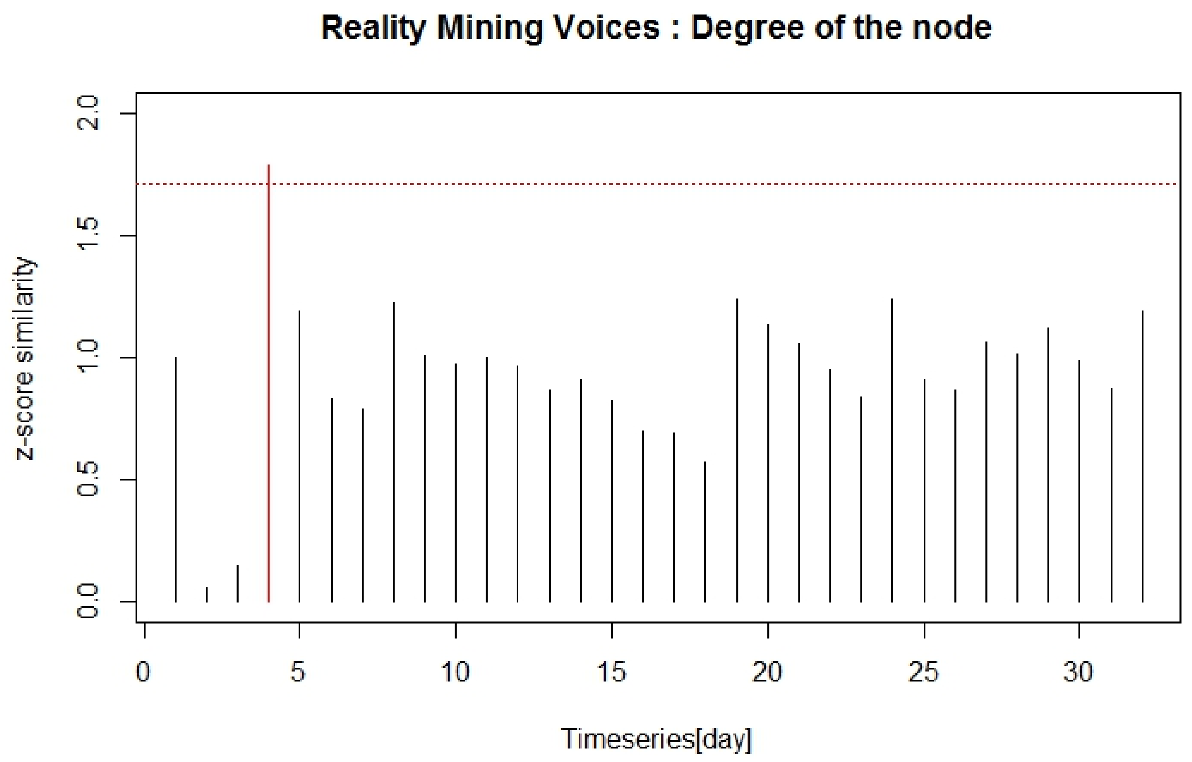
Program Output:

Following are the output graphs for the enron graph. Horizontal dotted red line indicates the upper threshold for detecting an anomaly. Anomalous nodes are represented by red lines while normal nodes by black lines.

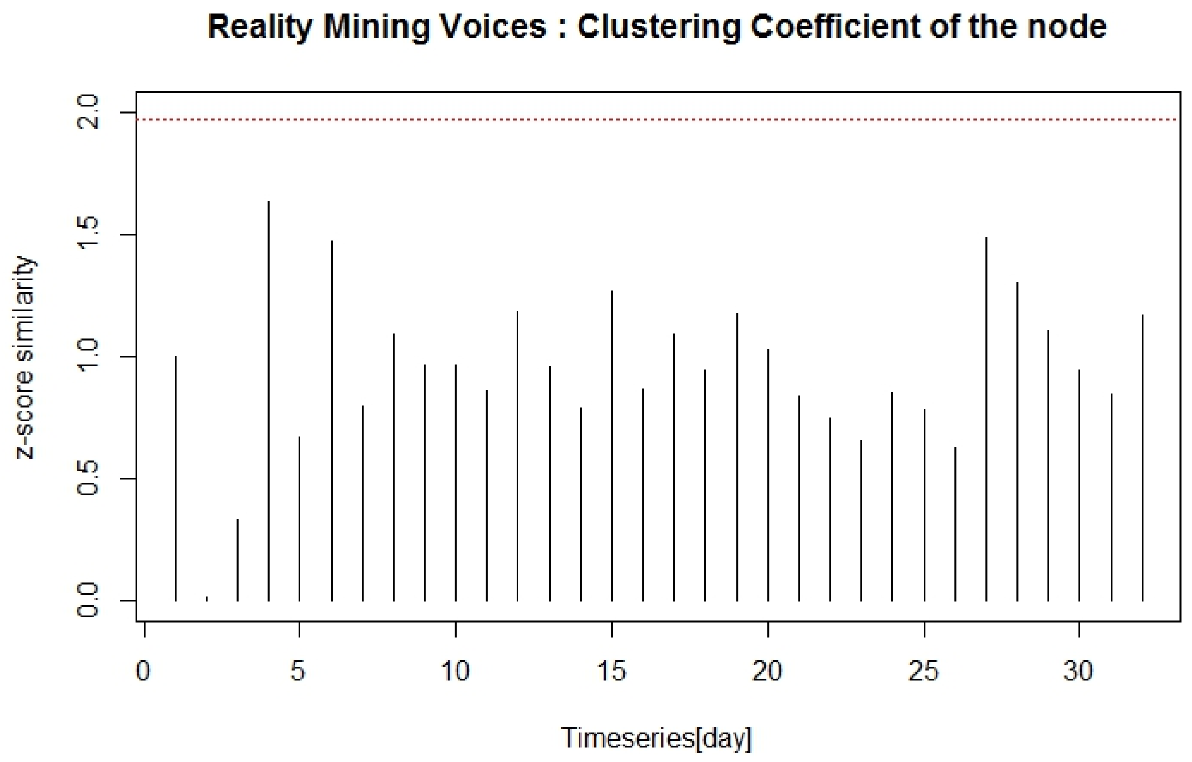



Enron Graph : Clustering Coefficient of the node
z-score similarity
Timeseries[day]
th = 1.596
0.0
0.5
1.0
1.5
2.0
0
200
400
600

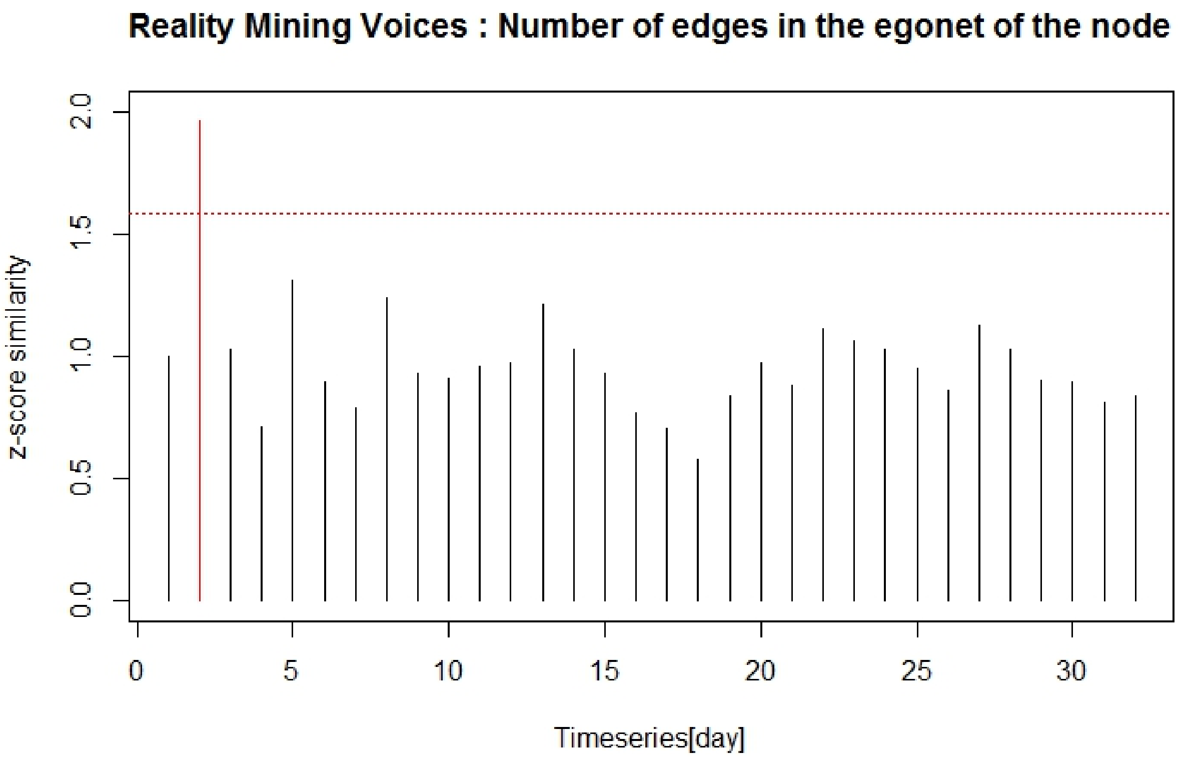

Enron Graph : Number of edges in the egonet of the node
z-score similarity
Timeseries[day]
th = 1.691
0.0
0.5
1.0
1.5
2.0
0
200
400
600



Reality Mining Voices : Degree of the node
z-score similarity
0.0
0.5
1.0
1.5
2.0
0
5
10
15
20
25
30
Timeseries[day]



Reality Mining Voices : Clustering Coefficient of the node
z-score similarity
0.0
0.5
1.0
1.5
2.0
0
5
10
15
20
25
30
Timeseries[day]



Algorithm Analysis & Findings:

1. **How could your algorithm be improved in order to be more robust and possibly identify more anomalies while not producing false positives?**
   - The threshold obtained using the moving range average over the similarity score(z-score) can be tuned according to the dataset provided to detect optimal number of anomalies, such that false positives are minimized. Also another improvement that can be done is taking different window sizes. User behavior can be different on weekdays and different on weekends. So windows can be adjusted such that we compare behavior of the weekdays with past behavior on weekdays and same with weekends. This approach may improve the algorithm and make it more robust and possibly identify more anomalie while not producing false positives.

2. **Could it be improved in terms of time or memory complexity?**
   - This algorithm generates a correlation matrix between every pair of nodes for each time window, which a big overhead in terms of computation time as well as memory. So an alternative technique that can do the job of correlation matrix can be devised to save on time and memory complexity.

3. **What kind of anomalies might your algorithm have trouble detecting? Why?**
   - As the algorithm compares the current user behavior with the past user behavior in a the given time window, it can efficiently recognize the contextual anomalies. But detecting **point anomalies** might be troublesome. For instance, if two anomalies lie in the same time window, those anomalies will be detected as a single anomaly and it would be difficult to narrow down as to which users/nodes caused these anomalies.

4. **Is your algorithm parameterized?**
   - No, this algorithm is not parameterized.The key parameter threshold is calculated in the algorithm based on the given graphs and user behavior.

5. **Randomized? Does it start from a seed set of vertices? How does this affect the performance?**
   - No. As the algorithm works on the time varying graphs, all the nodes in the graph are to be considered and in a particular time order to detect anomalies in the timeseries data. So randomization is not required for this algorithm. As this algorithms not randomized, the results are perfectly deterministic or stochastic.

6. **Report how your algorithm performed on the provided data sets. Consider: what percentage of the total number of input graphs did your algorithm identify as anomalous?**
   - Performance of the algorithm on different datasets based on the percentage of anomalous point/nodes/agents detected are:

     - Enron Graph : 0.7% (~6 out of 772)
     - Reality Mining Voices Graph : 2.9% (1 out of 34)

7. **What does this say about the sensitivity of the algorithm?**
   - Though anomaly detection of the algorithm may attain a high accuracy, there may be some errors due to some of the ignored factors such as the users behavior on the weekends, or public holidays or vacation leave,etc. So this may affect the sensitivity of the algorithm. So algorithm may be sensitive to some of such noisy user behavior.

8. **Are the detected time points very near each other or very spread out?**
   - The detected anomalous time points in the graph are spread out in the time series which determines some events or change points based on user behavior.

9. **Compare an anomalous timepoint to a normal time point. What differences do you see in the graph structure?**
   - Consider the feature, Degree of the node, for comparison of an anomalous time point to normal time point. It is clearly observed that the degree of the node 107 in the 4th graph(anomalous) of reality mining voices is very different from the degree of node 107 in

the other past graphs say graph 1(normal) which detects the change in behavior. Graph structure of the anomalous point shows major changes according to user behavior compared to past normal time points.

References:

1. Akoglu, Leman, and Christos Faloutsos. "Event Detection in Time Series of Mobile Communication Graphs." (n.d.): n. pag. Web.
2. "EgoNet." *Wikipedia*. Wikimedia Foundation, 11 Jan. 2014. Web. 01 Nov. 2014.
3. "Clustering Coefficient." *Wikipedia*. Wikimedia Foundation, 24 Oct. 2014. Web. 01 Nov. 2014.
4. "Package ‘egonet’." (n.d.): n. pag. Web.
5. "Home - RStudio." *RStudio*. N.p., n.d. Web. 01 Nov. 2014.