Paper:
EVENT DETECTION IN TIME SERIES OF MOBILE COMMUNICATION GRAPHS

Leman Akoglu*, Christos Faloutsos
Carnegie Mellon University
Pittsburgh, PA, 15213

Algorithm:

This paper proposes a solution to detect change points in a time varying graphs. Change points are the state where many nodes deviates from their normal behavior. Algorithm uses a set of features of the graph. First step is to extract the feature data for set of nodes in the time series. Next algorithm builds a correlation matrix for the feature data showcasing the correlation of behavior between all pairs of nodes in the graph over a certain period of time. Next algorithm extracts the behavior vector of the graph in the form of principal eigen vector and compares it with the past behavior window from previous time windows. For this comparison the algorithm uses a dot product between vectors. In case the current behavior is significantly different than the past behavior then the current time window is flagged as anomalous.

Detailed Steps:

- **Feature Extraction from nodes**: Paper uses several features of a node in the graph. Each node is represented as set of features namely in-degree, out-degree, in-weight, out-weight, number of neighbours,number of reciprocal neighbors ,number of triangles ,average in-weight, average out-weight, maximum in-weight, maximum out weight and maximum weight ratio on reciprocated edges egonet. The current implementation uses set of sub features namely degree of the node, clustering coefficient of the node and number of edges in the egonet of the node.
- **Change Point Detection:**After feature extraction we have data for T*N*F where T is number of days , N is the number of nodes in the time varying graph and F is the feature set. The algorithm takes slice of this data for one of the features so that we have data T*N for one of the features chosen. As a next step we take the subset of the T*N for window size W where W<<T to get W*N. On this data we calculate the pearson correlation matrix between these time series vectors for W days. The algorithm ideally by experiments uses W=7.Next we slide the window down by one tick or day and compute the correlations over the next window W time ticks. This step is repeated till the end of the data. Then we calculate the principal eigenvector for each of the correlation matrix.
- **Metric to score Time Points for Anomaly:** At a time t we find the average eigenvectors r(t-1)for last W' eigenvectors and then find the cross product between the principal eigenvector at t u(t). We repeat this for all eigenvectors. For each of the cross product we calculate Z score as z = 1 – u(t).r(t-1). Higher Z value indicates a change point and is flagged accordingly after comparing against threshold.

Implementation Details

In this implementation of the algorithm in R language window size W is chosen as 7 and W' as 5.

1)Algorithm reads the data from the input files .The format of input files is assumed to be in the from <day>_<graph_type>.txt. There is one file per each day. The first line in the file consists of number of edges present in the day's data space separated by the number of vertices in the entire graph.
2)Using the data read from the input file for each day we calculate the feature data for the day's graph. Feature can be set as Cluster Coefficient, Degree and number of edges present in the egonet.
3)We convert the day's edge list to the graph using igraph library. For Cluster coefficient we use the function transitivity from igraph package in R language. For calculating the degree we use degree function from the igraph package. We find the egonet for each node using 'ego.extract' function from the sna package.
4)We save the feature data for each day in a list so that we can have access to any day's feature data at O(1).
5)We take 7 days data and calculate the correlation matrix using 'cor' function from the stats package. Once we have the correlation matrix we calculate the principal eigen vector using the Eigen function.
6)Next we remove the earliest day's data from the 7 days data and add the next day's data to form a new vector of 7 days data and repeat the step 5.We do this till all the days data are covered.
7)Once we have the eigenvector ready for each correlation matrix the next step is to calculate the dot product between eigenvectors.
8)Once we have the eigenvector ready for each correlation matrix the next step is to calculate the dot product between eigenvectors.
9)Once we have the eigenvector ready for each correlation matrix the next step is to calculate the dot product between eigenvectors. The dot product is calculated between average of last 5 principal Eigen vectors and the current Eigen vector.
10)From the dot product of Eigen vectors we calculate the Z score given by
1- u(t)r(t-1). This forms the basis for detecting anomalies.
11)We calculate the threshold as Median + 3*Moving Range average and compare the z score results against this. Anything that is above this is treated as anomaly.
12) The anomaly points are stored in an output file. The first line consists of the number of anomaly points followed by the actual data points. Each point represents the first day of the week that may contain the anomaly.

How could the algorithm be improved in order to be more robust and possibly identify more anomalies while not reducing false positives?

- Algorithm can be extended further to do a deep analysis to figure out the points that contribute to the anomaly instead of detecting the week that has the anomaly.
- Algorithm needs to have sufficient day's data to yield good results. We can come up with optimizations or techniques to handle smaller size data.
- Initial few set of correlation matrices and last few set of correlation matrices have less significance on the algorithm since we use a sliding window. Hence we can make sure to account for their significance too.
- The algorithm assumes that initial few vectors depict a normal behavior which can become problematic incase that itself is an anomaly.

Could it be improved in terms of time and memory complexity?

- We can have a parallelized version of the algorithm to calculate the correlation matrix since each correlation calculation step is independent of each other. This can greatly reduce the time complexity required for a large graph.
- We can have parallel reading from the input files to gain faster execution.
- Algorithm considers data for each day and calculates the feature data for each day's data. After calculation it stores the feature data for all the days. We can have improvements to make sure we efficiently store the feature data so that memory is not wasted. One of the ways to do that is by storing the feature data for each day only for the vertices that are present for that day. Currently data is stored for all the global vertices though most of that values are 0.
- Currently the algorithm stores the principal Eigen vectors for all the correlation matrices ahead of time. This can be improved to calculate the principal eigen vectors as required on the fly thus reducing memory space required.

What kind of anomalies could your algorithm have trouble detecting? Why?

- When there is a small data set in terms of number of days there is a possibility that anomalies can go undetected since window size of 7 can mask the anomalies that are present
- If anomalies exist in the first few days then it can go undetected since while calculating the z score we start from the 5th day principal Eigen vector as detected by W'.

Is your algorithm parameterized?

- The algorithm is parameterized. These are,
    - Window Size, W : This is the number of days that we aggregate to calculate the correlation matrix. This is the size of the sliding window.
    - W": This represents the number of eigenvectors that are required to calculate typical eigenvector to derive the z score eventually.

By experiments it is seen that best results are obtained for W=7 and W'=5s

Is the algorithm Randomized?
NO. The algorithm is not randomized since it follows an order in execution steps when it comes to the input time series data.

Does it start from a seed set of vertices? how does it affect your performance?
NO. Algorithm doesn't start from a seed set of vertices. Algorithm always considers vertices from starting days in the order.

What percentage of the total number of input graphs did your algorithm identify as anomalous?

Percentage Of anomalies			
DataSet	Degree	Clustering Coefficeint	Edges in the egonet
Enron	0.56955	0.4317585	0.5787402
Enron Noempty	0.56248	0.4457995	0.598774
Reality mining voices	0.77778	1.555556	0.74584

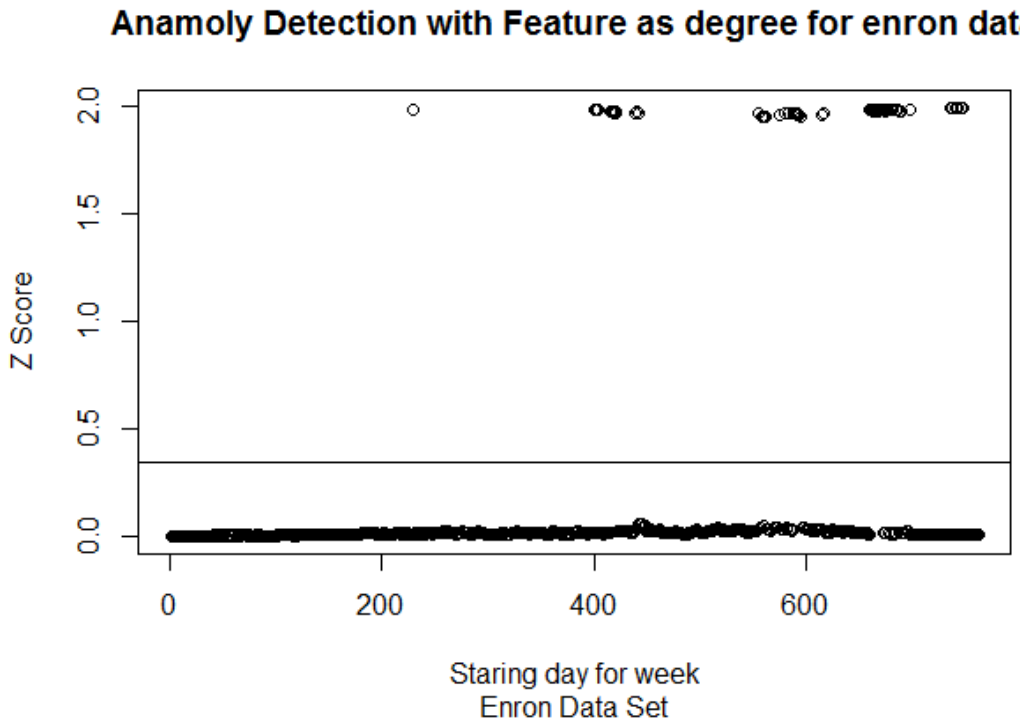
What does it say about sensitivity of the algorithm?

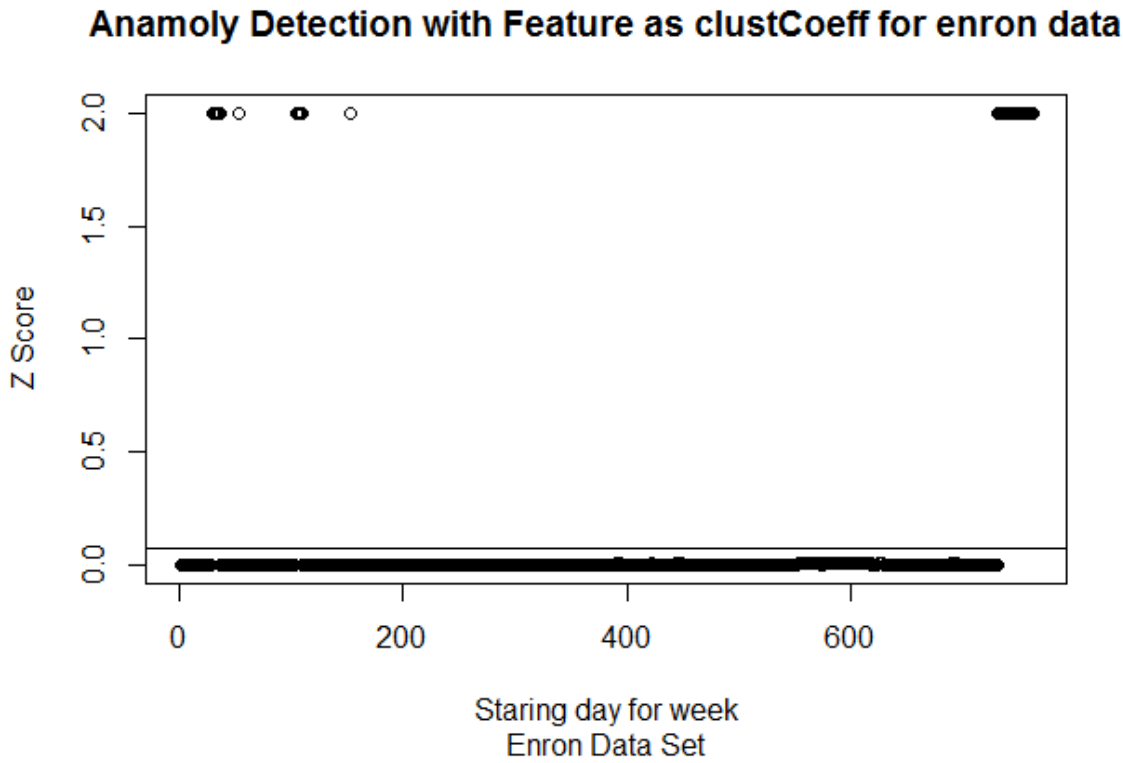
- The algorithm is sensitive to size of the data set since a smaller data set with W as 7 will aggregate the points together masking the anomaly
- Algorithm is also sensitive to W. With W=7 and W=7 algorithm performs better as compared to W=12 as mentioned in the paper .This is because as window size is increased more data falls under the window.

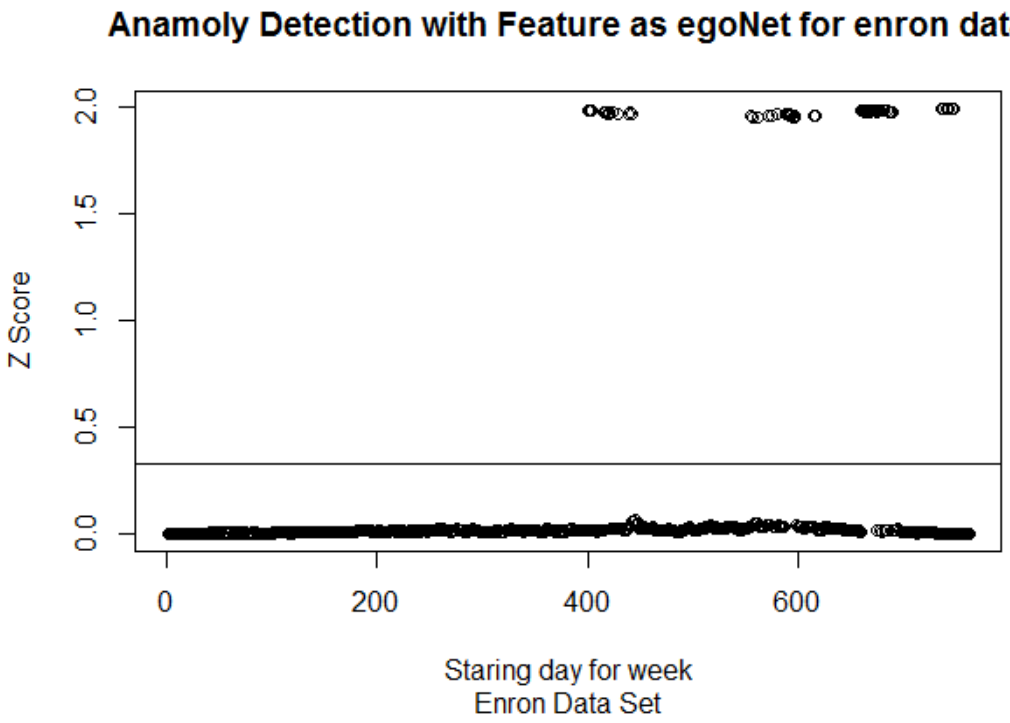
Are the detected time points vary near each other or very spread out?
They are spread out.

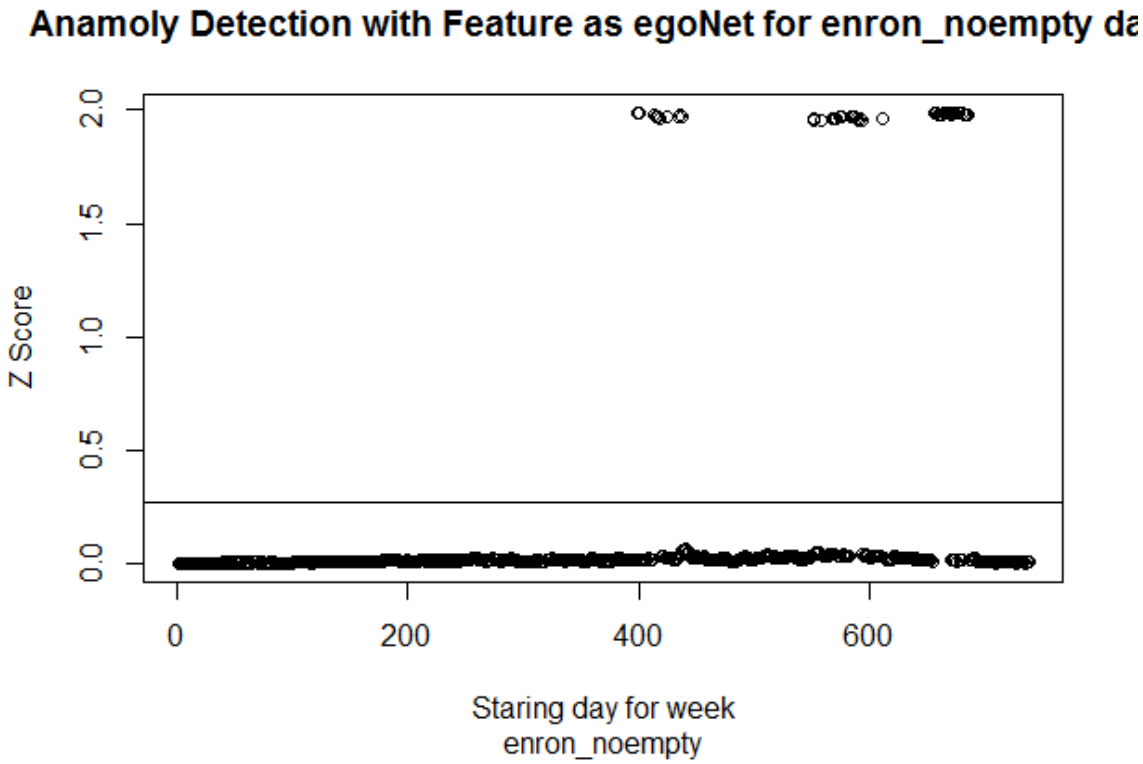
Compare the anomalous time points to normal time points?

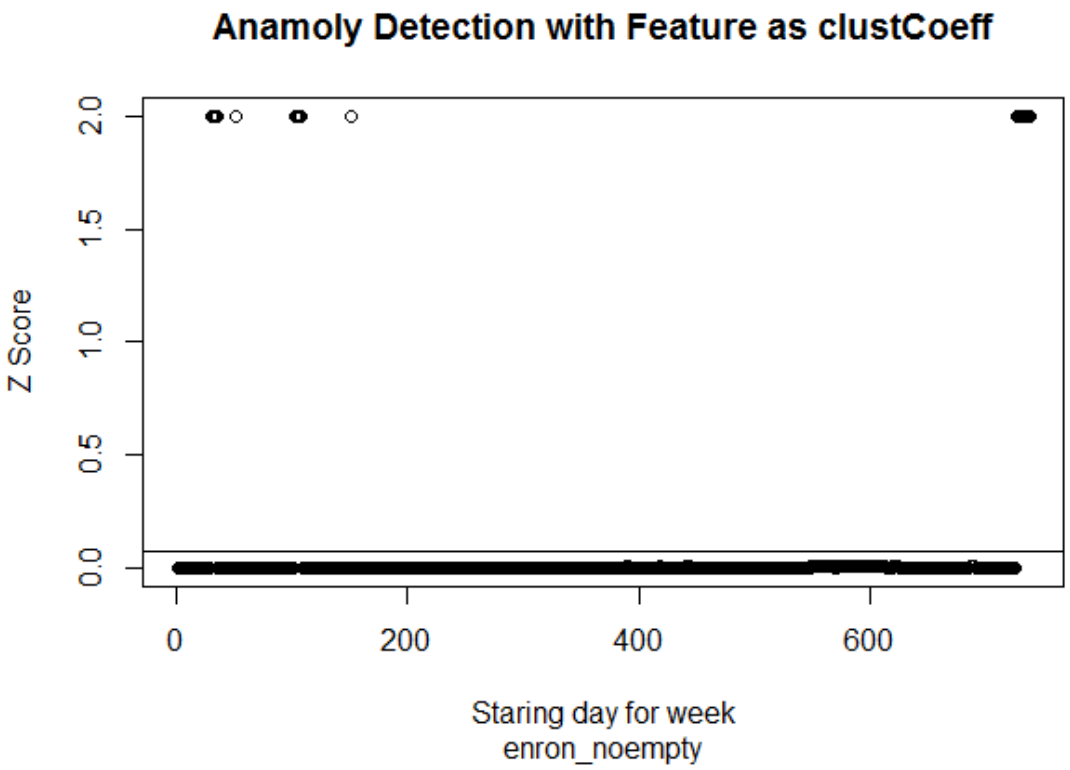
- Refer to the plots to compare the anomalous points in the time series. Note that each data points refer to anomaly in the 7 days window starting at that data point(W+day). Anomaly can exist in the interval [current_day, current-day+6] window.

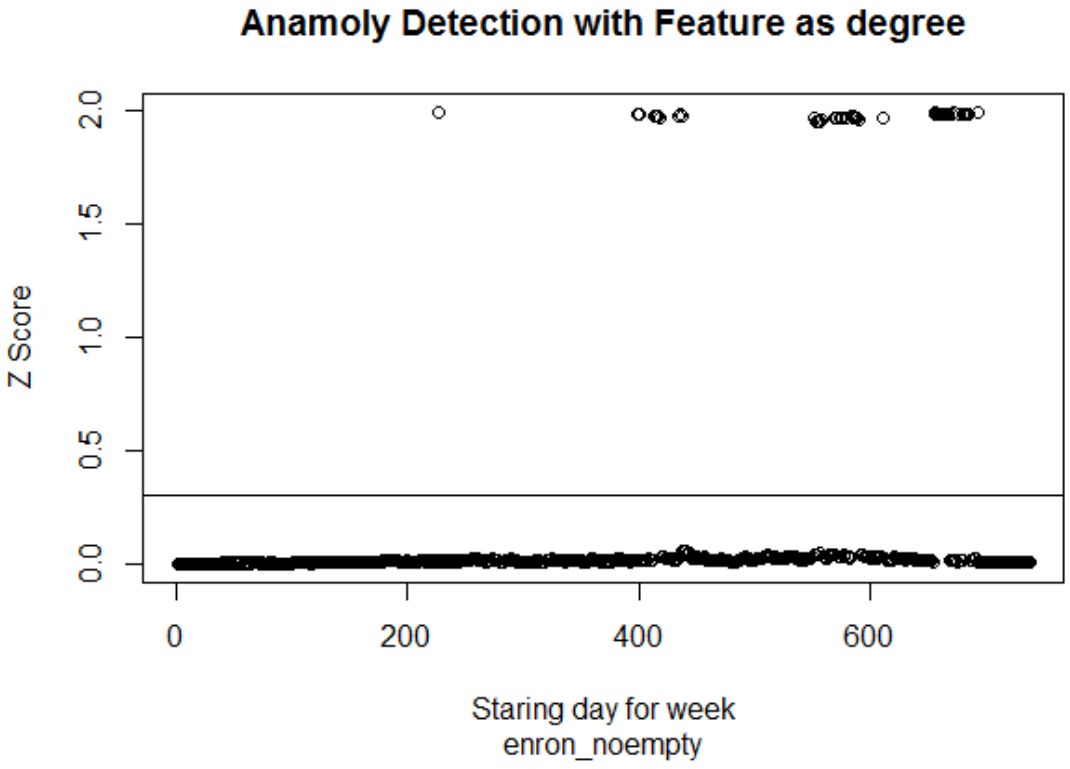

Anamoly Detection with Feature as degree for enron data
Z Score
0.0
0.5
1.0
1.5
2.0
0
200
400
600
Staring day for week
Enron Data Set

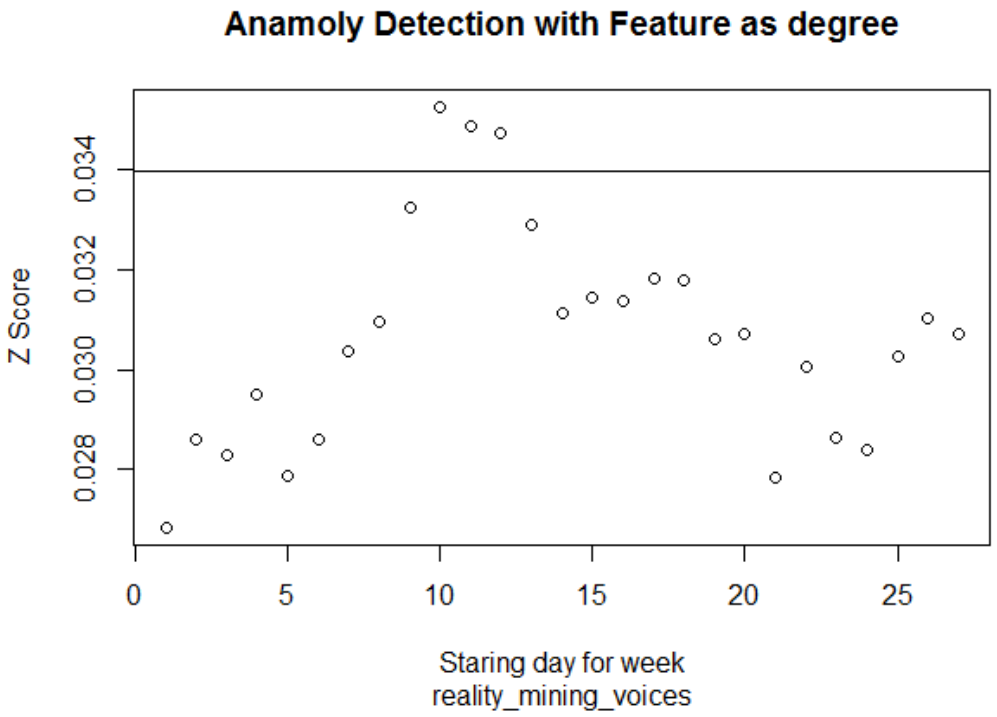

Anamoly Detection with Feature as clustCoeff for enron data
Z Score
0.0
0.5
1.0
1.5
2.0
0
200
400
600
Staring day for week
Enron Data Set

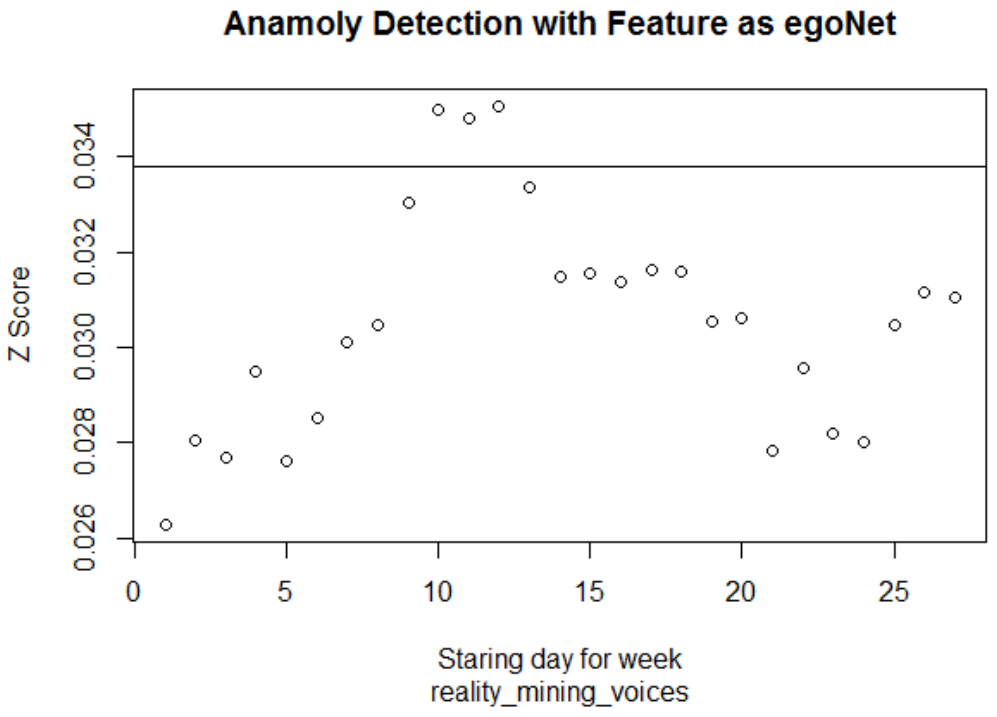

Anamoly Detection with Feature as egoNet for enron data
Z Score
0.0
0.5
1.0
1.5
2.0
0
200
400
600
Staring day for week
Enron Data Set

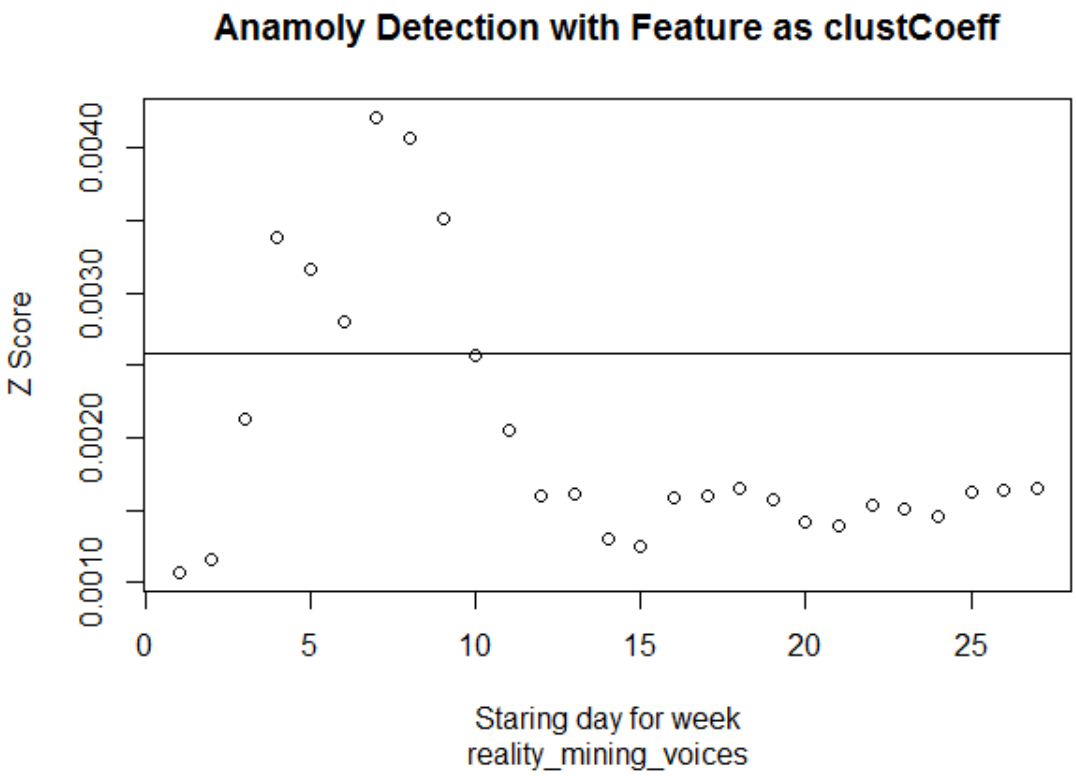

Anamoly Detection with Feature as egoNet for enron_noempty da
Z Score
0.0
0.5
1.0
1.5
2.0
0
200
400
600
Staring day for week
enron_noempty


Anamoly Detection with Feature as clustCoeff
Z Score
0.0
0.5
1.0
1.5
2.0
0
200
400
600
Staring day for week
enron_noempty


Anamoly Detection with Feature as degree
Z Score
0.0
0.5
1.0
1.5
2.0
0
200
400
600
Staring day for week
enron_noempty


Anamoly Detection with Feature as degree
Z Score
0.028
0.030
0.032
0.034
0
5
10
15
20
25
Staring day for week
reality_mining_voices


Anamoly Detection with Feature as egoNet
Z Score
0.026
0.028
0.030
0.032
0.034
0
5
10
15
20
25
Staring day for week
reality_mining_voices



What differences do you see in graph structure?

- All the anomalous points are way above the threshold point clearly differentiating the anomalous points. Z scores for anomalous data points are high compared to normal data points.