# Setup the graph and curve it

We just utilized the edge list file to setup the graph using function read.graph(), the graph is connected and the diameter is 8. Figure 1.1 shows the distribution of degree of this graph.

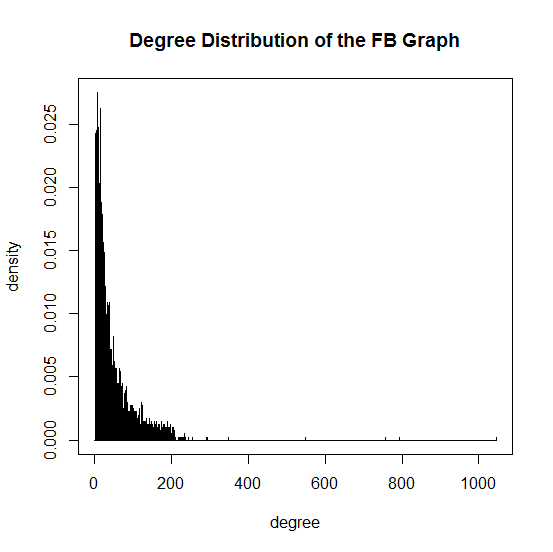


Figure1.1 Degree Distribution of generated graph

As for the fitting curve, we use the stat\_smooth() function to generate the statistics model. After studying the shape of the curve, we determine to try four models, which are y ~ log(x), y ~ I(1/x\*a) + b\*x, y ~ I(1/x\*a)+b and y ~ I(exp(1)^(a + b \* x) respectively. And as we can see from figure 1.2, the curve of y ~ I(exp(1)^(a + b \* x) fits best.

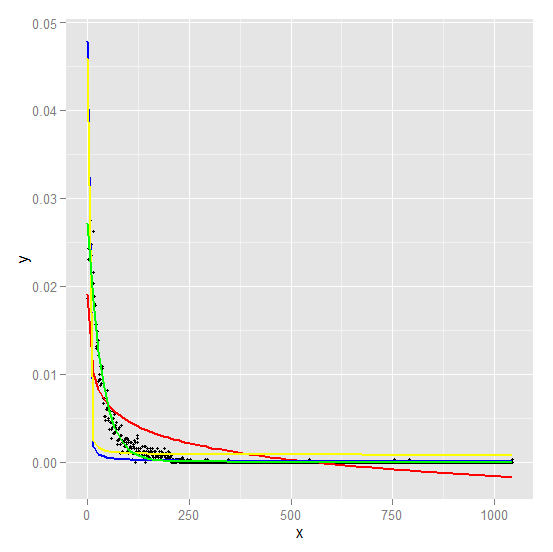


Figure1.2 Fitting curves with different models

The conclusion of the best model:

Formula: y ~ I(exp(1)^(a + b \* x)) where a=-3.594 and b=-0.029

Residual standard error: 0.0006342 on 1043 degrees of freedom

Number of iterations to convergence: 15

Achieved convergence tolerance: 7.799e-07

And the MSE (total mean squared error) is 4.016458e-07 and average degree is 43.69101.

# Generate the personal network of node 1

We generate the personal network which is the subgraph of the whole graph with node 1 and its neighbors, we can see from figure 2.1 that all the nodes (except for node 1) share a mutual friend node 1. The total edges of this personal network are 2866 and the total nodes are 348.

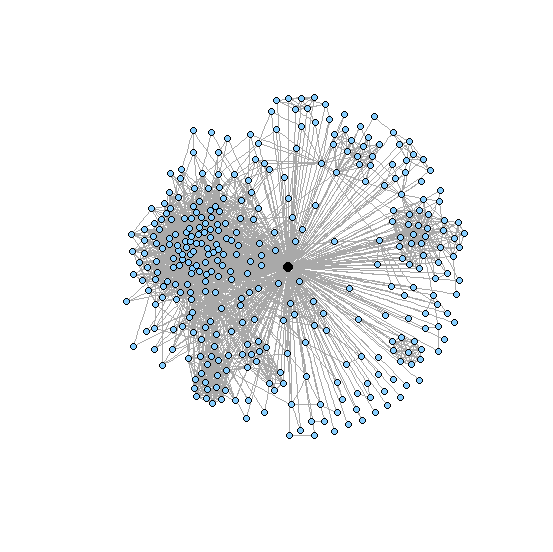


figure 2.1 personal network of node 1

# Analyze the community structure of core nodes

We just implemented a for loop and totally found 40 such core nodes in the graph and the average degree of such nodes are 279.375. We selected node 1 to analyze, which is a core node, in details.

Figure 3.1 shows the community structure of personal network 1 using fast-greedy algorithm; figure 3.2 shows the community structure using edge-betweenness algorithm and figure 3.3 shows the community structure using infomap algorithms respectively.

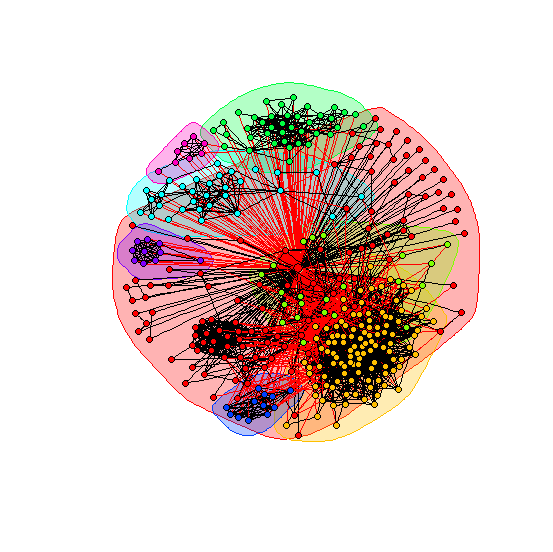


Figure 3.1 Community Structure using fast-greedy algorithm

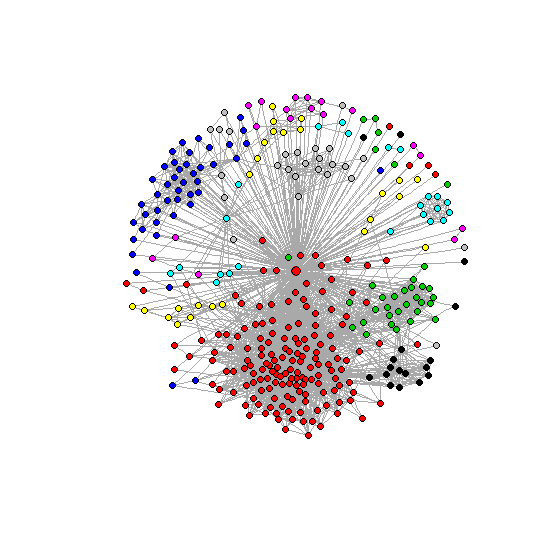


Figure 3.2 Community Structure using edge-betweenness algorithm

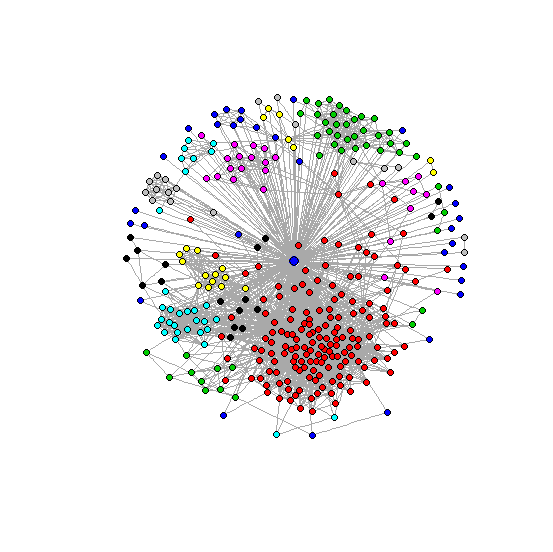


Figure 3.3 Community Structure using infomap algorithm

We see that, though determined using different algorithms, the communities in different graph have some apparent overlap, which mean these community structures have distinguish features. And it can be seen that edge-betweenness algorithm tends to break the graph into more partitions than other two algorithms.

And the modularity of 3 algorithms are 0.4131014, 0.3533022 and 0.3891185 respectively.

The following figure shows community structure of the 3 algorithms.

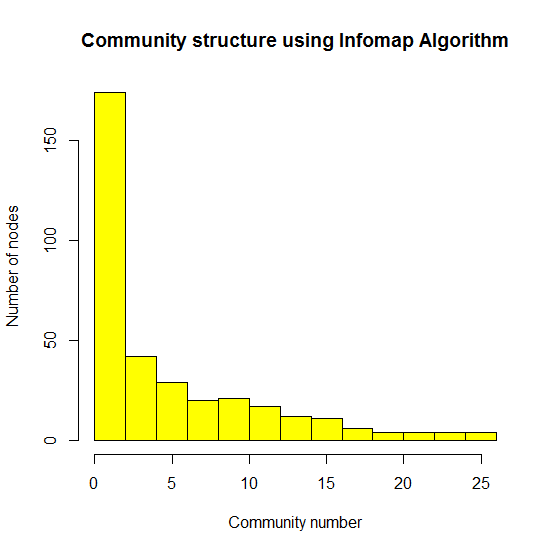
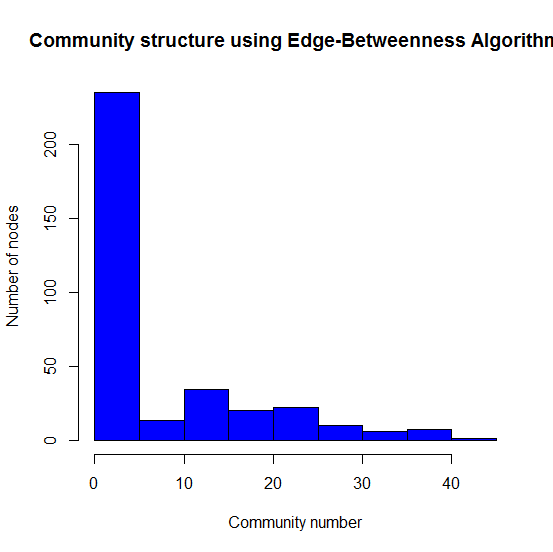
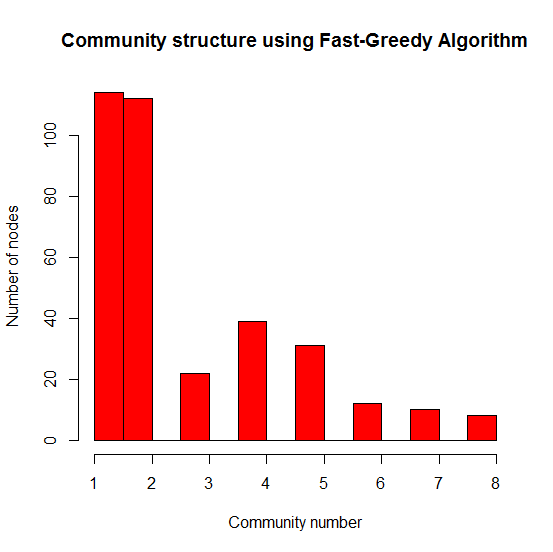
  

Figure 3.4 community structure of the 3 algorithms

# Analyze the subgraph without core nodes

Firstly we generated a graph without the core node 1 and then checked the community structure again as done above using three different algorithms. The results are shown in figure 4.1-4.3.

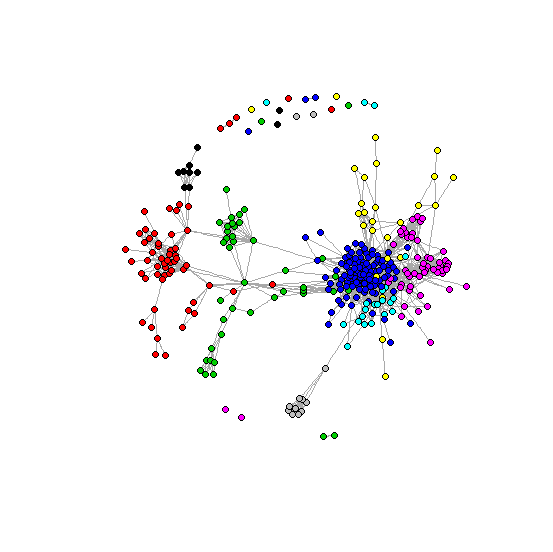


Figure 4.1 Community Structure using fast-greedy algorithm

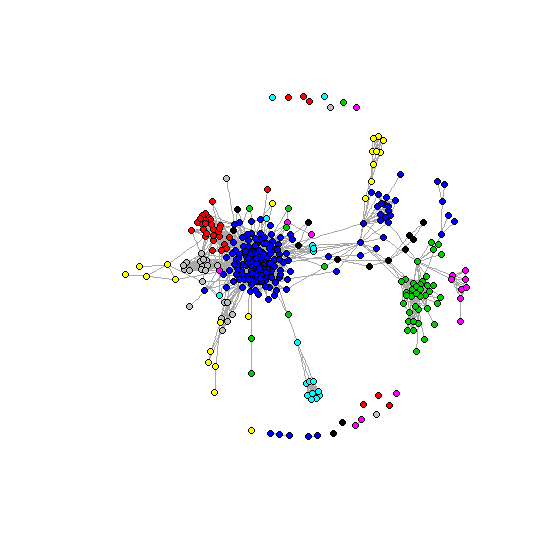


Figure 4.2 Community Structure using edge-betweenness algorithm

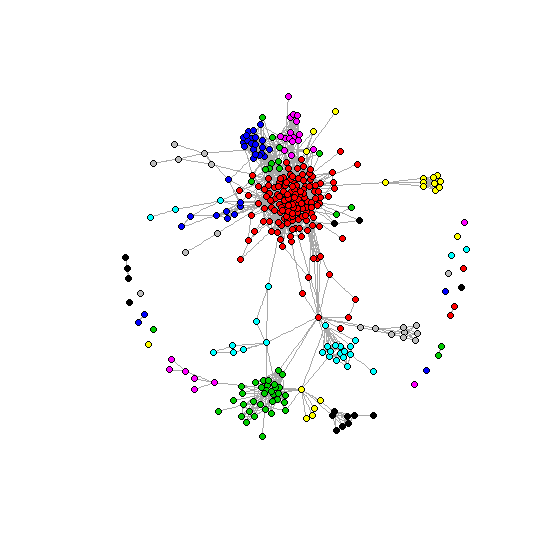


Figure 4.3 Community Structure using infomap algorithm

Looking further into those community structures, we can find out that though they are structured without the core node, the partitions are actually similar. And this can be testified by examine the modularity of structures of problem 3 and problem 4.

And the modularity of 3 algorithms are 0.4418533, 0.4161461 and 0.4180077 respectively.

The difference of modularity is about 10% between two parts.