# An Example of Social Network Analysis with R using Package igraph

The data to analyze is Twitter text data of @RDataMining used in [the example of Text Mining](http://www.rdatamining.com/examples/text-mining), and it can be downloaded as file “termDocMatrix.rdata” at [the Data webpage](http://www.rdatamining.com/data). Putting it in a general scenario of social networks, the terms can be taken as people and the tweets as groups on LinkedIn, and the term-document matrix can then be taken as the group membership of people. We will build a network of terms based on their co-occurrence in the same tweets, which is similar with a network of people based on their group memberships.

At first, a term-document matrix, termDocMatrix, is loaded into R. After that, it is transformed into a term-term adjacency matrix, based on which a graph is built. Then we plot the graph to show the relationship between frequent terms, and also make the graph more readable by setting colors, font sizes and transparency of vertices and edges.

### Load Data

> # load termDocMatrix  
> load(“data/termDocMatrix.rdata”)  
> # inspect part of the matrix  
> termDocMatrix[5:10,1:20]

Docs

Terms 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

data 1 1 0 0 2 0 0 0 0 0 1 2 1 1 1 0 1 0 0 0

examples 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

introduction 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

mining 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0

network 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1

package 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

### Transform Data into an Adjacency Matrix

> # change it to a Boolean matrix  
> termDocMatrix[termDocMatrix>=1] <- 1  
> # transform into a term-term adjacency matrix  
> termMatrix <- termDocMatrix %\*% t(termDocMatrix)  
> # inspect terms numbered 5 to 10  
> termMatrix[5:10,5:10]

Terms

Terms data examples introduction mining network package

data 53 5 2 34 0 7

examples 5 17 2 5 2 2

introduction 2 2 10 2 2 0

mining 34 5 2 47 1 5

network 0 2 2 1 17 1

package 7 2 0 5 1 21

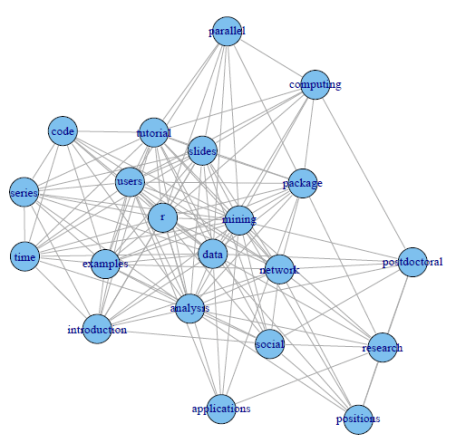
### Build a Graph

Now we have built a term-term adjacency matrix, where the rows and columns represents terms, and every entry is the number of co-occurrences of two terms. Next we can build a graph with graph.adjacency() from package igraph.

> library(igraph)  
> # build a graph from the above matrix  
> g <- graph.adjacency(termMatrix, weighted=T, mode = “undirected”)  
> # remove loops  
> g <- simplify(g)  
> # set labels and degrees of vertices  
> V(g)$label <- V(g)$name  
> V(g)$degree <- degree(g)

### Plot a Graph

> # set seed to make the layout reproducible  
> set.seed(3952)  
> layout1 <- layout.fruchterman.reingold(g)  
> plot(g, layout=layout1)



A different layout can be generated with the first line of code below. The second line produces an interactive plot, which allows us to manually rearrange the layout. Details about other layout options can be obtained by running ?igraph::layout in R.

> plot(g, layout=layout.kamada.kawai)  
> tkplot(g, layout=layout.kamada.kawai)

### Make it Look Better

Next, we will set the label size of vertices based on their degrees, to make important terms stand out. Similarly, we also set the width and transparency of edges based on their weights. This is useful in applications where graphs are crowded with many vertices and edges. In the code below, the vertices and edges are accessed with V() and E(). Function rgb(red, green, blue, alpha) defines a color, with an alpha transparency. We plot the graph in the same layout as the above figure.

> V(g)$label.cex <- 2.2 \* V(g)$degree / max(V(g)$degree)+ .2  
> V(g)$label.color <- rgb(0, 0, .2, .8)  
> V(g)$frame.color <- NA  
> egam <- (log(E(g)$weight)+.4) / max(log(E(g)$weight)+.4)  
> E(g)$color <- rgb(.5, .5, 0, egam)  
> E(g)$width <- egam  
> # plot the graph in layout1  
> plot(g, layout=layout1)

