Investment Events Analysis

Wenbo(Robin) Liu

```
library(igraph)
```

```
library(ggplot2)
df1<-read.csv('Funding_events_7.14.csv',stringsAsFactors = FALSE)
df2<-read.csv('Funding_events_7.14_page2.csv',stringsAsFactors = FALSE)
colnames(df2)=colnames(df1)
raw_data<-rbind(df1,df2)
```

First we'd want to organize our data. In order to have more control over any potential problems I wrote a function to do so. But typically as.Date should work fine.

```
change_date<-function(date){
  n<-nchar(date)
  two_dig<-as.numeric(substr(date,(n-1),n))

  if(two_dig<10){
    four_dig<-paste0('200',two_dig)

  } else if(two_dig>=10 && two_dig<=15){
    four_dig<-paste0('20',two_dig)
  } else {
    four_dig<-paste0('19',two_dig)
  }
  date_str<-paste0(substr(date,1,(n-2)),four_dig)
  return(date_str)
}
test.date<-character(nrow(raw_data))
for(i in 1:nrow(raw_data)){
  test.date[i]<-change_date(raw_data$Deal.Date[i])
}
```

```
#Sort the data by date.
raw_data.sorted<-raw_data
raw_data.sorted$Deal.Date<-as.Date(test.date,format = "%m/%d/%Y")
raw_data.sorted<-raw_data.sorted[order(raw_data.sorted$Deal.Date),]
initial.month<-raw_data.sorted$Deal.Date[1]
initial.month<-12*as.numeric(substr(initial.month,1,4))+
  as.numeric(substr(initial.month,6,7))
```

Process edge lists, each month’s investor, investors from all time, and compute the coreness with decay at this step. We also store igraph object of jun 1991 for future use.

```
#Keep track of # of months
count_month<-0;
#initialize the dataframe that stores decaying network coreness
month_cor_df<- data.frame('month'=0,'coreness'=0)
#The overall edge list: x1,x2 = investor name
#x3 = the current number of dated month
#x4 = the last time we saw an edge. This is useful
#     because we ignore any edge if the last time
#     we saw it was 120 month(10 years) ago or longer.
edge_list<-data.frame('x1'=character(),'x2'=character(),'x3'=numeric(),
                      'x4'=numeric())
#A list that documents each month's investor.
investor_grand_list<-list()
#An array that documents all unique investor.
unique_invester.arr<-c()
#A number that helps us to keep track of while loop process.
nraw<-nrow(raw_data)
#strings we splited and would be treated as investors that are not actual inv
estors.
#We'll deal with them later.
bad_string<-c('inc','inc.','llc','llc.','ltd','ltd.')
#This function will be applied at each row,creating edges.
edges<-function(x,y,z){
```

```
  related_investor<-trimws(unlist(strsplit(as.character(x[11]),split=',')),'b
oth')


  if(length(related_investor)>1){

    edg<-as.data.frame(t(combn(related_investor,2)))

    edg<-cbind(edg,rep(y,nrow(edg)))

    edg<-cbind(edg,rep(z,nrow(edg)))

    return(edg)

  }


}

g_1991<-NA

q4.month<-1991*12+6

#At each iteration, takes the first date(year-month) and grab all rows that s
hare the same

#date. It processes it, update investor information, edge_list and calculates
 decay-network

#coreness for each month, and remove the rows from the dataframe afterward. T
herefore at the

#next iteration we'll always have a new,more current date at the first row si
nce the dataframe

#was sorted in the last step.

while(nrow(raw_data.sorted)>0){

  #tell us how many rows are left

  keep.track<-nraw-nrow(raw_data.sorted)

  #update how many months we've seen

  count_month<-count_month+1

  #date to be processed.

  year_month<-substr(raw_data.sorted$Deal.Date[1],1,7)

  #Translate year-month to a integer(12*year+month), easier for calculation i
n the future.

  current.month<-12*as.numeric(substr(raw_data.sorted$Deal.Date[1],1,4))+as.n
umeric(substr(raw_data.sorted$Deal.Date[1],6,7))

  #rows involved with this month that we need to process

  y_m_index<-which(substr(raw_data.sorted$Deal.Date,1,7)==year_month)

  year_month_df<-raw_data.sorted[y_m_index,]
```

```
  #keep track of investors appeared in this month
  investor_month_list<-paste0(year_month_df$Investors,',')

  text_list<-character()
  for(i in 1:length(investor_month_list)) {
    temp_vector <- unlist(strsplit(investor_month_list[i],split = ','))
    text_list<-append(text_list,temp_vector)
  }

  #list of investors appeared this month
  investor_month_list<-unique(sort(trimws(text_list,which='both')))

  #for each row within this month's data frame, find out edges
  #and recored the appeared month.
  temp_edges<-apply(year_month_df,1,edges,count_month,current.month)
  if(!is.null(temp_edges)){
    temp_edges<-do.call(rbind,temp_edges)
    colnames(temp_edges)<-c('x1','x2','x3','x4')
  }

  #update edge_list
  edge_list<-rbind(edge_list,temp_edges)

  #Update our overall investors and each month's investor.
  unique_invester.arr<-unique(sort(c(unique_invester.arr,investor_month_lis
t)))
  investor_grand_list[[count_month]]<-investor_month_list

  #Build a vertex dataframe using overall investors
  vertices<-data.frame('v'=unique_invester.arr)
  #Determine which edges within edge list are seen within 10 year.
  keep<-which((current.month-edge_list$x4)<=120)
  #make the igraph object with appropriate edges and all the investors we've
seen so far.
  g_temp<-graph.data.frame(edge_list[keep,],vertices = vertices,directed = FA
LSE)
```

```
  g_temp<-simplify(g_temp,remove.multiple = TRUE,edge.attr.comb = 'max')
  #The way we splited 'investor' from the raw data leaves us with some bad st
rings that
  #are not really investors. We need to delete them before we move on.
  for(i in bad_string){
    while(!is.na(match(i,tolower(V(g_temp)$name)))>0){
      bad_index<-match(i,tolower(V(g_temp)$name))
      g_temp<-delete_vertices(g_temp,bad_index)
    }
  }
  #Store igraph object of jun1991, therefore
  #we don't have to go through this process again.
  if(current.month==q4.month){
    g_1991<-g_temp
  }
  #Calculate mean coreness(with decay)
  mean_corness<-mean(coreness(g_temp,'all'))

  #store month along with its associated coreness.
  m_c.df<-data.frame('month'=current.month,'coreness'=mean_corness)
  month_cor_df<-rbind(month_cor_df,m_c.df)
  #Remove data we've processed
  raw_data.sorted<-raw_data.sorted[-y_m_index,]

}
#First row is c(0,0) and has no value to us.
month_cor_df<-month_cor_df[-1,]
```

consider all data, from 1981 to 2014. Who is in the center? It depends on what aspect we consider

```
max.degree<-V(g_temp)$name[which.max(centralization.degree(g_temp)[[1]])]
max.closeness<-V(g_temp)$name[which.max(closeness(g_temp))]
max.eigen<-V(g_temp)$name[which.max(eigen_centrality(g_temp)[[1]])]
max.between<-V(g_temp)$name[which.max(betweenness(g_temp))]
print(paste0('The investor who has the most collaboration with others is: ',m
ax.degree))
```

```
[1] "The investor who has the most collaboration with others is: Intel Capita
l"
```

```
print(paste0('The investor who takes the least stepes to reach other investor
s is: ',max.closeness))
```

```
[1] "The investor who takes the least stepes to reach other investors is: Int
el Capital"
```

```
print(paste0('The investor who is the most influential: ',max.eigen))
```

```
[1] "The investor who is the most influential: Intel Capital"
```

```
print(paste0("The investor who is the most important 'bridge' between others:
 " ,max.between))
```

```
[1] "The investor who is the most important 'bridge' between others: Intel Ca
pital"
```

Therefore Intel Capital is the center of the network in many aspects, with no doubt.

Calculate the average distance between investor:

```
average.path.length(g_temp)
```

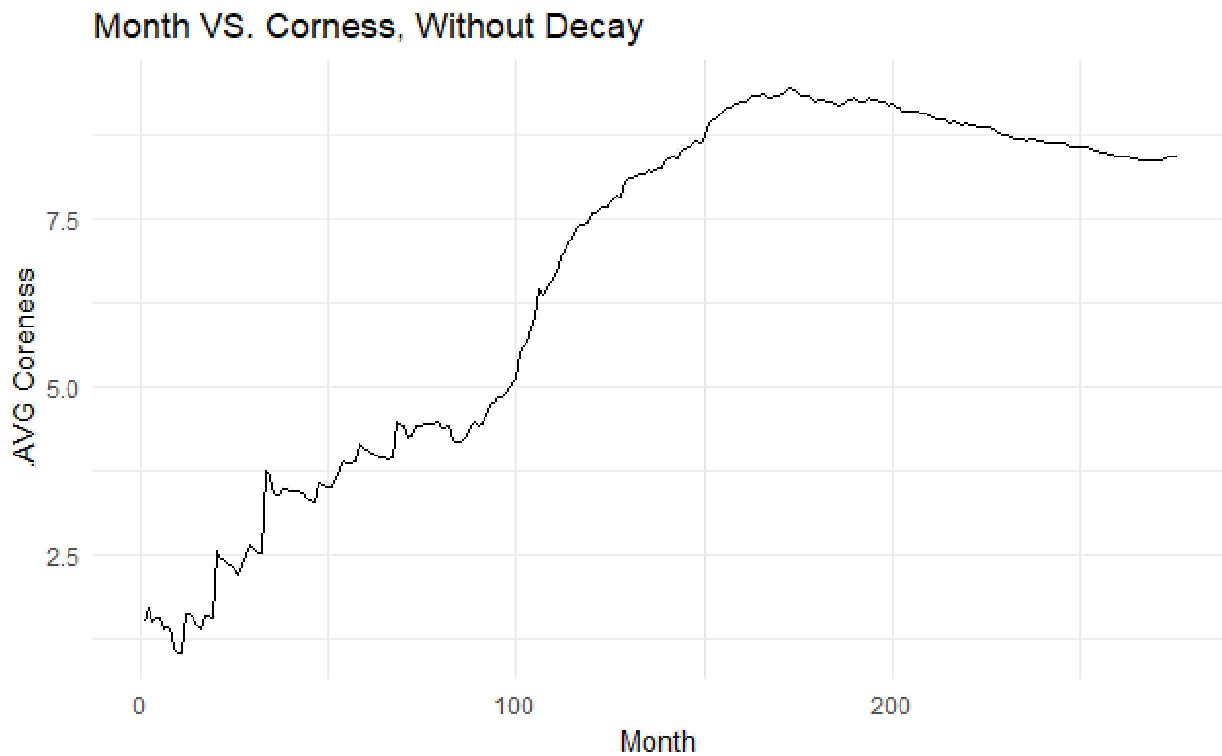
```
[1] 3.523307
```

This is quite high, and investors are likely to need to bypass a ‘center investor’ in order to reach to others; in this case the ‘center investor’ is very likely to be intel capital.

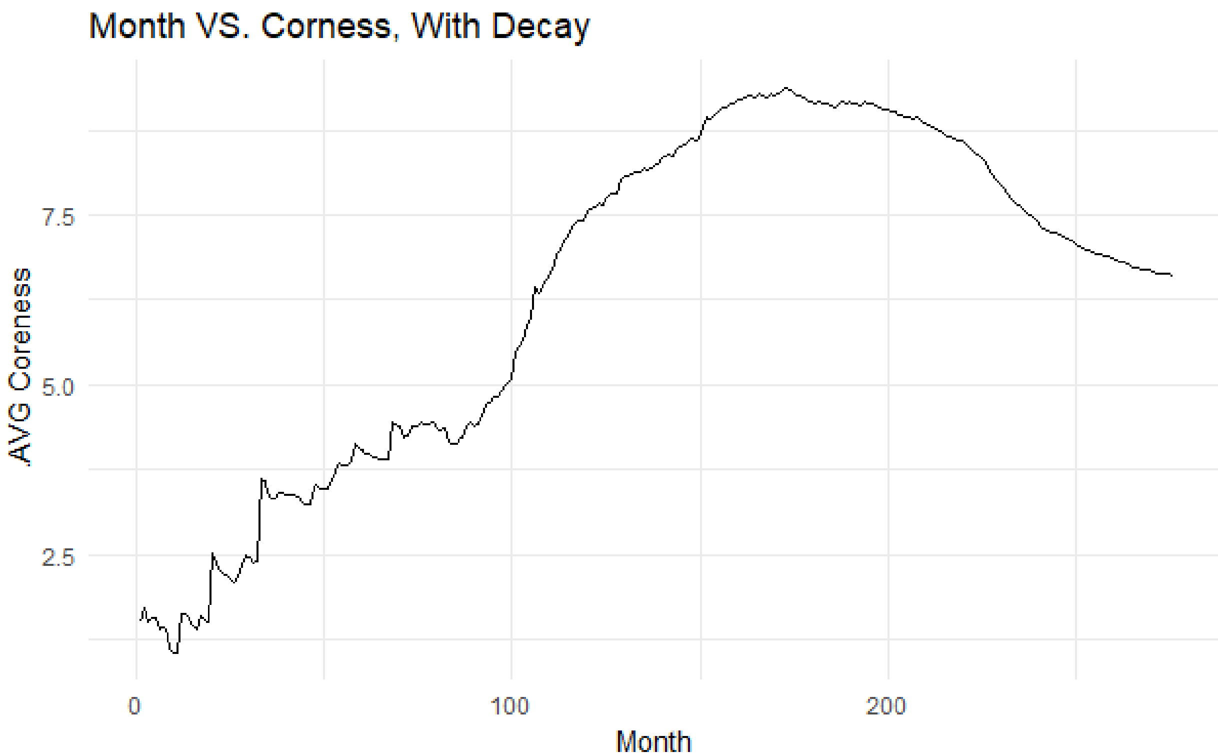
Since we’ve already have the edge lists of all time as well as each month’s investor information, the computation is much easier.

```
#determine how many months there are.

months<-unique(month_cor_df[,1])

#create a dataframe to store non-decay network monthly coreness.

month_cor_df.nd<-data.frame('month'=numeric(),'coreness'=numeric())

#Create a overall graph, and for each month we create a small graph and union

#with the overall graph.

temp_frame<-data.frame('x1'=character(),'x2'=character())

overall_graph<-graph.data.frame(temp_frame, directed = FALSE)

#At each iteration, create a graph with edges & vertices seen at that month,
then union with
```

```
#the overall graph.
for (i in 1:length(months)){
  vertices.nd<-investor_grand_list[[i]]
  temp_df<-edge_list[which(edge_list[,3]==i),]
  temp_g<-graph.data.frame(temp_df,vertices = vertices.nd, directed = FALSE)
  temp_g<-simplify(temp_g,remove.multiple = TRUE)
  overall_graph<-union(overall_graph,temp_g)
  #Again, we delete false-investors before we make calculations.
  for(i in bad_string){
    while(!is.na(match(i,tolower(V(overall_graph)$name)))>0){
      bad_index<-match(i,tolower(V(overall_graph)$name))
      overall_graph<-delete_vertices(overall_graph,bad_index)
    }
  }
  #calculate monthly coreness for non-decay network, and store it.
  mean.core<-mean(coreness(overall_graph))
  temp_result<-data.frame('month'=months[i],'coreness'=mean.core)
  month_cor_df.nd<-rbind(month_cor_df.nd,temp_result)
}
g<-ggplot()+geom_line(aes(x=c(1:nrow(month_cor_df.nd)),y=month_cor_df.nd$coreness))+ggtitle('Month VS. Corness, Without Decay')+theme_minimal()+labs(x='Month',y='.AVG Coreness')
g
```




```
g<-ggplot()+geom_line(aes(x=c(1:nrow(month_cor_df)),y=month_cor_df$coreness))
+ggtitle('Month VS. Corness, With Decay')+theme_minimal()+labs(x='Month',y='.
AVG Coreness')

g
```



This plot is not very different from the previous plot. This indicates that investors are constantly investing with same partners and connected to the same other investors.

Let’s also take a look at a clustering approach. For the June 1991 network, use the iterative correlation clustering method, which takes the correlation of the original adjacency matrix and keeps taking the correlation of this result until it converges to a matrix of 1s and ???1s, which splits the network into clusters of cliques.

```
adj<-as.matrix(get.adjacency(g_1991))
zeros<-c(1,4:6,8,11,14,17,19,20,27:30,33,35)
adj<-adj[-zeros,-zeros]
while(sum(which(abs(adj)<1))>0){
  adj<-cor(adj)
}
adj.p<-adj
adj.p[adj.p<1]=0
g.p<-graph.adjacency(adj.p,'undirected',diag = FALSE)
```

```
plot(g.p)
```

The fact that average distance between investors is greater than 3 indicates that the investors exhibit a core-periphery structure. Instead of having many neighbors, investors need to bypass some ‘center investors’ in order to reach to others. The last plot, two similar-sized clusters can be an piece of evidence as well: there are two equivalent class of investors exist on both sides of ‘center investers’; they need to have a center invester in order to connect to each other.