Assignment3 of FIT3152

(1). In this question, I collected 15 news items from the ABC News website, covering three different topics. Within each topic, I specifically chose five articles. From these selected articles, I extracted sentences(with more than 100 words) that were of particular interest to me. These sentences were then copied and pasted into a newly created plain text file located in the designated text folder. These files will be used to do text analysis later.

(2). To create the corpus, I began by setting the working directory to the correct location. Next, I cleared the workspace and imported all the necessary packages. Since my files are already in the TXT format, there was no need for conversion. I simply used the folder that contains these files and added all of them to the corpus using the Corpus() function. Since I used the directory as the source, I utilized the DirSource() function. Below is a list of all the files that were added to the corpus.

```
> #then read the folder which contain all the text file we needed
> cname = file.path(".", "text_folder") #
> docs = Corpus(DirSource((cname))) #add all the file in to Corpus
> print(summary(docs)) #print all the file we have here
              Length Class            Mode
Business1.txt 2      PlainTextDocument list
Business2.txt 2      PlainTextDocument list
Business3.txt 2      PlainTextDocument list
Business4.txt 2      PlainTextDocument list
Business5.txt 2      PlainTextDocument list
Health1.txt   2      PlainTextDocument list
Health2.txt   2      PlainTextDocument list
Health3.txt   2      PlainTextDocument list
Health4.txt   2      PlainTextDocument list
Health5.txt   2      PlainTextDocument list
Science1.txt  2      PlainTextDocument list
Science2.txt  2      PlainTextDocument list
Science3.txt  2      PlainTextDocument list
Science4.txt  2      PlainTextDocument list
Science5.txt  2      PlainTextDocument list
```

(3). In this part, our objective is to create a Document-Term Matrix (DTM). To accomplish this, we undertake the following steps:

1. Tokenization: We divide the text into individual words. During this process, we replace any occurrences of "-" (all kind dash) with spaces, remove numbers and punctuation, and convert all words to lowercase. This process converts continuous text into discrete words, making subsequent text processing more convenient and efficient.

2. Word Filtering: Our next step involves removing stop words in English and deleting any remaining spaces or blank lines. Filtering out irrelevant words can reduce data noise and improve the effectiveness of subsequent analysis.

3. Stemming: We unify words by reducing them to their root form. In this case, we utilize English-based stemming. Merge different morphological variants into the same root, thereby reducing interference caused by variations of words.

Having completed the preprocessing steps, we now proceed to create the DTM. To achieve this, we utilize the DocumentTermMatrix() function, which takes the pre-processed documents as input. Additionally, we analyse the term frequencies by calculating how often each term appears across all the documents. We sort the terms in ascending order of frequency and create a frequency table to examine the distribution. This allows us to determine the number of terms occurring a certain number of times and provides insights into the folder number and the count of different word tokens.

```
> freq[head(ord)] #less
   bureau      crisi      delta     econom  economist        gdp
        1          1          1          1          1          1
> freq[tail(ord)] #most
     will australian       cent        per       year       said
        9         10         11         11         18         22
```

We can see some words like ‘said’, ‘year’, ‘will’. They appear many times and might appear in every type of files. To improve the accuracy of cluster we need to do, I remove excluding certain terms which may be common on all topics and make dtm again to make dtm better.

```
> freq[head(ord)] #less
   bureau      crisi      delta     econom  economist        gdp
        1          1          1          1          1          1
> freq[tail(ord)] #most
       rate       chequ         age australian       cent        per
          8           8           9         10         11         11
```

Analysing this information aids in deciding how to handle sparse terms. In this case, we utilize the removeSparseTerms() function, setting a threshold of 0.75 to retain approximately 20 word tokens, as specified in the question. The reason why is 0.75 is below: if we set is as 0.8 it will have 60 tokens, if set as 0.7 only left 5 tokens. So, 0.75 (with 24 tokens) is most properly.

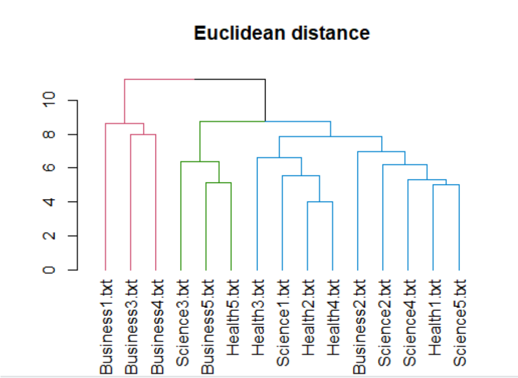
```
> dtms = removeSparseTerms(dtm, 0.75) #here we want only left about 20 tokens
> dim(dtms) #to show how many token after we doing it
[1] 15 24
> dtms = removeSparseTerms(dtm, 0.8) #here we want only left about 20 tokens
> dim(dtms) #to show how many token after we doing it
[1] 15 60
> dtms = removeSparseTerms(dtm, 0.7) #here we want only left about 20 tokens
> dim(dtms) #to show how many token after we doing it
[1] 15  5
```

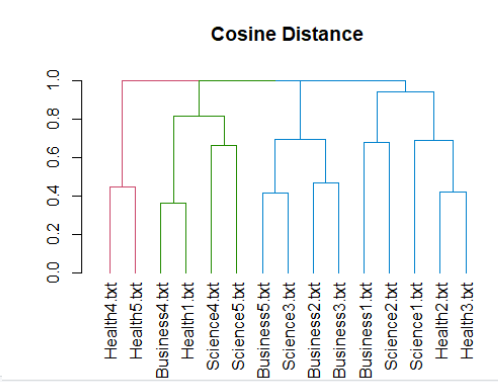
Then make dtms as the matrix and save as csv file, put it as appendix.

(4). In this part, I use both Euclidean distance and cosine distance.

```
> fit_cos

Call:
hclust(d = dtm_cos)

Cluster method   : complete 
Distance         : cosine 
Number of objects: 15 

> fit

Call:
hclust(d = distmatrix, method = "ward.D")

Cluster method   : ward.D 
Distance         : Euclidean 
Number of objects: 15 
```

here ‘fit’ is for Euclidean distance, ‘fit_cos’ is for Cosine distance.





In this progress I as.dendrogram() function to make graph more visual and I use color_branches() function to colour the clusters of clustering results by k=3(because I have 3 type of topics), then we can see the result more clearly.

Also, I use the cutree function to see it, it is print out the cluster for each file assigned to.

Below is for Euclidean distance: like what we do in colouring it, we set k=3 and sort it(for convenience). Then we can see how these file assign to different clusters. Here we can see that Business1,3,4 is in the same cluster class which all files are from same topic so they should in the same cluster. Then second cluster class contain Business2, Health1,2,3,4 and Science1,2,4,5 which situation happen might because these files contain the similar word, for example, similar words, expressions, or contexts are used. Also, there may be some crossover or overlap between science news and health news. For example, certain news stories may involve content that is both scientific and health. We can see there is a business file in this cluster might because some news talk about Health-related economic issues. Lastly, we see cluster 3 have files Business3, Health5, Science3. All these files from different topic but they are in the same cluster class. But when I read the text file I found, they are not similar news, so the reason that happens is there may be noise or outliers in the data set that cause texts on different topics to be incorrectly grouped into the same cluster.

```
> cutfit = cutree(fit, k = 3)
> cutfit #use to get more information about which cluster is each sample assigned to
Business1.txt Business2.txt Business3.txt Business4.txt Business5.txt   Health1.txt
            1             2             1             1             3             2
  Health2.txt   Health3.txt   Health4.txt   Health5.txt  Science1.txt  Science2.txt
            2             2             2             3             2             2
 Science3.txt  Science4.txt  Science5.txt
            3             2             2
> sort(cutfit)
Business1.txt Business3.txt Business4.txt Business2.txt   Health1.txt   Health2.txt
            1             1             1             2             2             2
  Health3.txt   Health4.txt  Science1.txt  Science2.txt  Science4.txt  Science5.txt
            2             2             2             2             2             2
Business5.txt   Health5.txt  Science3.txt
            3             3             3
```

Below is for Cosine distance: like what we do in colouring it, we set k=3 and sort it(for convenience). Then we can see how these file assign to different clusters. The cluster class 1 contain the Business1,2,3,5 , Health2,3 and Science1,2,3. Which might because Clustering algorithms or parameter Settings are not suitable for processing text data on different topics and might because they are all similar. Then cluster class 2 contain Business4, Health1, Science4,5. reason for this situation should be same as what we say before. Lastly cluster class 3 contain Health4 and Health5, they are all from same topic which means that is correct.

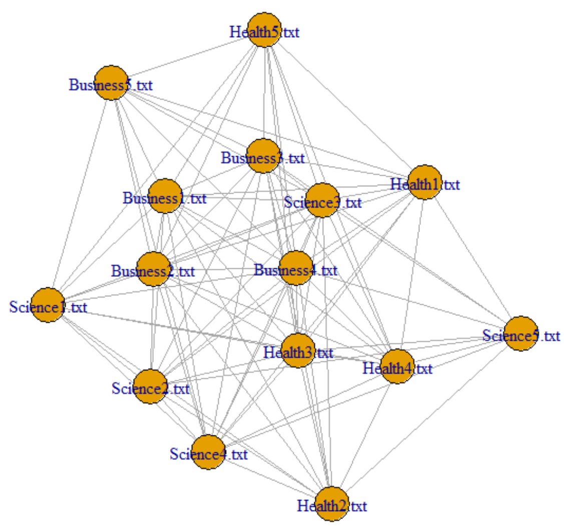
```
> cutfit_cos = cutree(fit_cos, k = 3)
> cutfit_cos #use to get more information about which cluster is each sample assigned to
Business1.txt Business2.txt Business3.txt Business4.txt Business5.txt   Health1.txt
            1             1             1             2             1             2
  Health2.txt   Health3.txt   Health4.txt   Health5.txt  Science1.txt  Science2.txt
            1             1             3             3             1             1
 Science3.txt  Science4.txt  Science5.txt
            1             2             2
> sort(cutfit_cos) # Output the cluster to which each document belongs
Business1.txt Business2.txt Business3.txt Business5.txt   Health2.txt   Health3.txt
            1             1             1             1             1             1
 Science1.txt  Science2.txt  Science3.txt Business4.txt   Health1.txt  Science4.txt
            1             1             1             2             2             2
 Science5.txt   Health4.txt   Health5.txt
            2             3             3
```

Last part in this question is give a quantitative measure of the quality of the clustering. Here I use Silhouette Coefficient and Calinski-Harabasz Index to prove it.

```
> library(fpc)
> #install.packages("fpc")
> #Silhouette Coefficient
> silhouette_score <- cluster.stats(distmatrix, cutfit)$avg.silwidth #should from -1 to 1, higher
value means better
> silhouette_score
[1] 0.06029111
> silhouette_score_cos <- cluster.stats(distmatrix, cutfit_cos)$avg.silwidth
> silhouette_score_cos
[1] -0.03370824
> #Calinski-Harabasz Index
> ch_index <- cluster.stats(distmatrix, cutfit)$ch #higher is better
> ch_index
[1] 2.180116
> ch_index_cos <- cluster.stats(distmatrix, cutfit_cos)$ch
> ch_index_cos
[1] 1.143047
>
```

Here we get Silhouette Coefficient for Euclidean distance is 0.06029111, it's Calinski-Harabasz Index is 2.180116. then for cosine distance the Silhouette Coefficient is -0.03370824 and Calinski-Harabasz Index is 1.143047. These values show us Euclidean distance cluster is better than cosine distance cluster, because it has higher value in both. Normally the cosine distance cluster should be better than Euclidean distance, but in this case Euclidean distance clustering may be a better fit for my dataset.

(5). In this part, we first use the code in lecture 12 to create network from DTM, then we can plot it out, here is the original network, then I will find the most important nodes in this network. For now, we cannot see any clear group and relationship, so we need to find the most important nodes in this network to show more.



To find the most important document/node. I will calculate the 'degree', 'closeness', 'eigenvector' and 'betweenness' of each document/node.

```
> #now start to find the most important document in the network
> d = as.table(degree(ByAbs)) #the degree of each document
> b = as.table(betweenness(ByAbs)) #the betweenness of each document
> c = as.table(closeness(ByAbs)) #the closeness of each document
> e = as.table(evcent(ByAbs)$vector) #the eigenvector of each document
> stats = as.data.frame(rbind(d,b,c,e))#combine the d,b,c,e together
> stats = as.data.frame(t(stats)) #change col to row
> colnames(stats) = c("degree", "betweenness", "closeness", "eigenvector") #assign correct name
```

I use this code to make all the 'degree', 'closeness', 'eigenvector' and 'betweenness' in one data frame and then assign correct name for it. Then we show it, we get:

```
> stats
              degree betweenness  closeness eigenvector
Business1.txt     13   1.0333333 0.04347826   0.8322429
Business2.txt     13   1.1722222 0.04000000   0.6368610
Business3.txt     13   0.5833333 0.04166667   0.9290360
Business4.txt     14   0.6818182 0.03703704   1.0000000
Business5.txt      9   1.3333333 0.03703704   0.5615587
Health1.txt       12   6.1171717 0.04545455   0.6369860
Health2.txt       11  14.1873737 0.05263158   0.3623024
Health3.txt       13   1.3611111 0.03703704   0.7412181
Health4.txt       12   4.1262626 0.04545455   0.5030347
Health5.txt       10   3.6575758 0.04166667   0.5105713
Science1.txt      11   6.8651515 0.04761905   0.3687200
Science2.txt      11   1.1666667 0.04166667   0.4731687
Science3.txt      14   0.0000000 0.03703704   0.8631944
Science4.txt      13  13.7469697 0.05263158   0.4187447
Science5.txt       9   2.6500000 0.04347826   0.4048547
> 
```

Then we sort by each of these conditions to show the most important one:

Start with betweenness:

```
> head(stats[order(-stats$betweenness),])
             degree betweenness  closeness eigenvector
Health2.txt      11   14.187374 0.05263158   0.3623024
Science4.txt     13   13.746970 0.05263158   0.4187447
Science1.txt     11    6.865152 0.04761905   0.3687200
Health1.txt      12    6.117172 0.04545455   0.6369860
Health4.txt      12    4.126263 0.04545455   0.5030347
Health5.txt      10    3.657576 0.04166667   0.5105713
```

Then closeness:

```
> head(stats[order(-stats$closeness),])
              degree betweenness  closeness eigenvector
Health2.txt       11   14.187374 0.05263158   0.3623024
Science4.txt      13   13.746970 0.05263158   0.4187447
Science1.txt      11    6.865152 0.04761905   0.3687200
Health1.txt       12    6.117172 0.04545455   0.6369860
Health4.txt       12    4.126263 0.04545455   0.5030347
Business1.txt     13    1.033333 0.04347826   0.8322429
```

Then eigenvector:

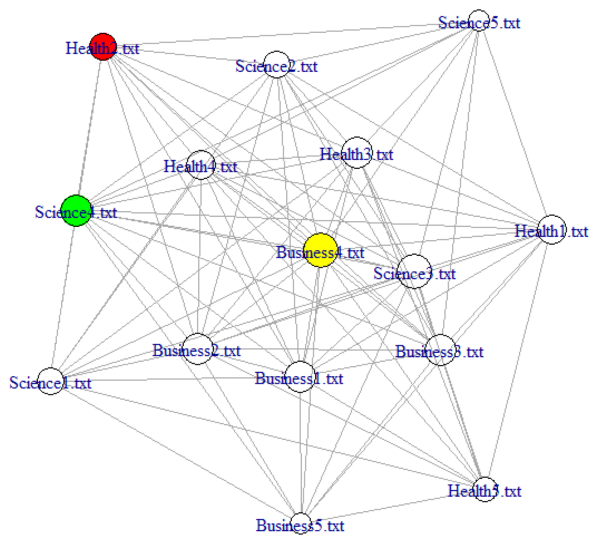
```
> head(stats[order(-stats$eigenvector),])
              degree betweenness  closeness eigenvector
Business4.txt     14   0.6818182 0.03703704   1.0000000
Business3.txt     13   0.5833333 0.04166667   0.9290360
Science3.txt      14   0.0000000 0.03703704   0.8631944
Business1.txt     13   1.0333333 0.04347826   0.8322429
Health3.txt       13   1.3611111 0.03703704   0.7412181
Health1.txt       12   6.1171717 0.04545455   0.6369860
```

Lastly degree:

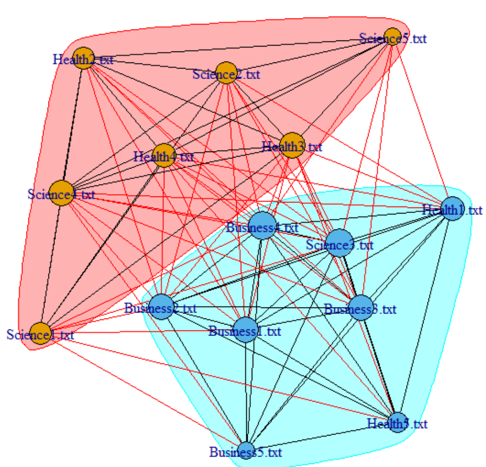
```
> head(stats[order(-stats$degree),],8)
              degree betweenness  closeness eigenvector
Business4.txt     14   0.6818182 0.03703704   1.0000000
Science3.txt      14   0.0000000 0.03703704   0.8631944
Business1.txt     13   1.0333333 0.04347826   0.8322429
Business2.txt     13   1.1722222 0.04000000   0.6368610
Business3.txt     13   0.5833333 0.04166667   0.9290360
Health3.txt       13   1.3611111 0.03703704   0.7412181
Science4.txt      13  13.7469697 0.05263158   0.4187447
Health1.txt       12   6.1171717 0.04545455   0.6369860
```

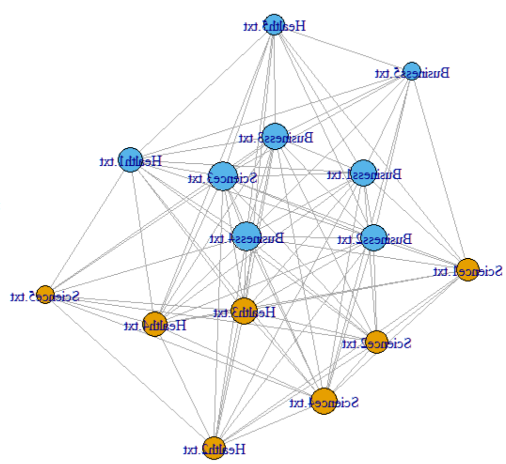
From the data provided above, it is evident that 'Business4.txt' has the highest degree (14), indicating that it is connected to the greatest number of edges and further reinforces its significance as one of the most important nodes. Additionally, 'Business4.txt' also holds the highest Eigenvector centrality (1), solidifying its importance in the network. Another node worth noting is 'Health2.txt', which possesses the highest betweenness (14.187374) and closeness (0.05263158) measures, making it a crucial node in the network. Moreover, 'Science4.txt' exhibits the highest closeness (0.05263158) and second highest betweenness (13.746970) and degree (13), thus emphasizing its significance as one of the most important nodes. In conclusion, the most important nodes in the network are 'Business4.txt', 'Health2.txt', and 'Science4.txt'.

Now I have highlighted the most important nodes in the graph using different colors. Health2.txt is shown in red, Business4.txt in yellow, and Science4.txt in green. Additionally, I have adjusted the size of the nodes based on their degrees, so nodes with higher degrees appear larger:



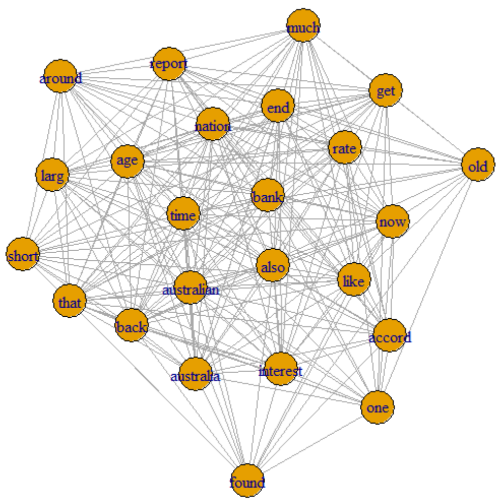
Also use community detection to plot graph can make you graph look more interesting, there are lots of way to do it, I just show two of these ways: 1. cluster_fast_greedy 2. Spinglass





On the left, we use the cluster_fast_greedy algorithm, and on the right, we use the spinglass algorithm. The cluster_fast_greedy algorithm divides the network into two communities based on the close correlation between nodes. On the other hand, the spinglass algorithm performs a similar task but divides the network into two communities. These are some interesting and useful ways to draw graph.

(6). In this part, we repeat all the activities as what we do in question 5. But this time is using tokens as the node to make network. Below is how this token-based network look like: For now, we cannot see any clear group and relationship, so we need to find the most important nodes in this network to show more.



Now is the time to find the most important Token/node. I will calculate the ‘degree’, ‘closeness’, ‘eigenvector’ and ‘betweenness’ of each Token/node.

```
> #now start to find the most important document in the network
> dT = as.table(degree(ByToken)) #the degree of each document
> bT = as.table(betweenness(ByToken)) #the betweenness of each document
> cT = as.table(closeness(ByToken)) #the closeness of each document
> eT = as.table(evcent(ByToken)$vector) #the eigenvector of each document
> statsT = as.data.frame(rbind(dT,bT,cT,eT))#combine the dT,bT,cT,eT together
> statsT = as.data.frame(t(statsT)) #change col to row
> colnames(statsT) = c("degree", "betweenness", "closeness", "eigenvector") #a
ssign correct name
```

I use this code to make all the ‘degree’, ‘closeness’, ‘eigenvector’ and ‘betweenness’ in one data frame and then assign correct name for it. Then we show it, we get:

```
> statsT
           degree betweenness  closeness eigenvector
accord         17   4.0588023 0.02777778   0.6077509
also           19   3.1639250 0.02777778   0.7298498
australia      20   2.9765873 0.02857143   0.6999637
bank           19   1.2317460 0.02564103   0.7605182
end            20   2.2504329 0.02631579   0.7943408
interest       20   6.2421356 0.03030303   0.6174357
much           19   8.0453102 0.03125000   0.5125959
nation         22   4.5436869 0.02857143   0.7645397
now            21  11.7051587 0.03125000   0.5895256
rate           20   9.1147186 0.03030303   0.5753601
australian     21   0.1428571 0.02325581   1.0000000
get            18   6.2298341 0.02857143   0.5129396
larg           19   9.3595238 0.03125000   0.5556666
one            15   4.5556277 0.02857143   0.3844733
report         20   4.5980519 0.02941176   0.7259420
that           21  11.8615079 0.03225806   0.5717414
time           21   1.3289683 0.02439024   0.8168008
age            21   6.5151515 0.02941176   0.7562870
around         17   8.1853175 0.02941176   0.4773544
back           21   8.1992063 0.03125000   0.5844751
found          14   2.4968615 0.02564103   0.4643355
short          16   5.7830087 0.02941176   0.4822561
old            14   4.0920635 0.02702703   0.3918768
like           19   4.1087662 0.02941176   0.5430664
```

Then we sort by each of these conditions to show the most important one:

Start with betweenness:

```
> head(statsT[order(-statsT$betweenness),]) #we only need to know the high mos
t so use head
        degree betweenness  closeness eigenvector
that        21   11.861508 0.03225806   0.5717414
now         21   11.705159 0.03125000   0.5895256
larg        19    9.359524 0.03125000   0.5556666
rate        20    9.114719 0.03030303   0.5753601
back        21    8.199206 0.03125000   0.5844751
around      17    8.185317 0.02941176   0.4773544
```

Then closeness:

```
> head(statsT[order(-statsT$closeness),])
         degree betweenness  closeness eigenvector
that         21   11.861508 0.03225806   0.5717414
much         19    8.045310 0.03125000   0.5125959
now          21   11.705159 0.03125000   0.5895256
larg         19    9.359524 0.03125000   0.5556666
back         21    8.199206 0.03125000   0.5844751
interest     20    6.242136 0.03030303   0.6174357
```

Then eigenvector:

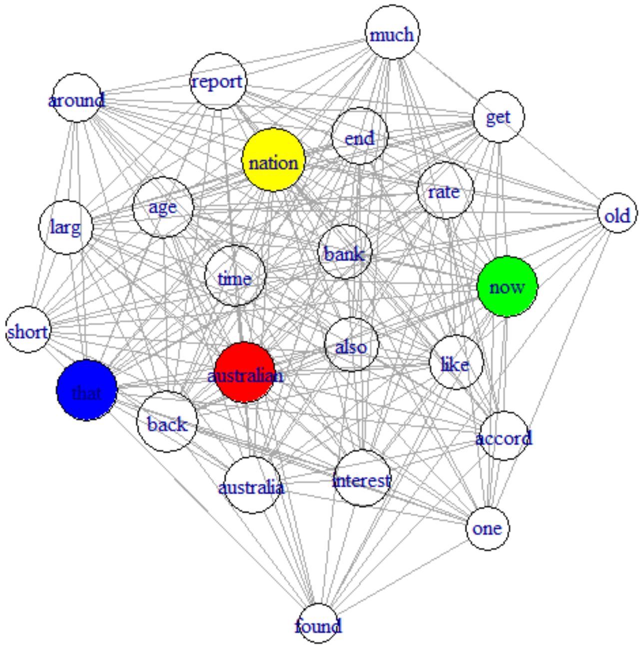
```
> head(statsT[order(-statsT$eigenvector),])
           degree betweenness  closeness eigenvector
australian     21   0.1428571 0.02325581   1.0000000
time           21   1.3289683 0.02439024   0.8168008
end            20   2.2504329 0.02631579   0.7943408
nation         22   4.5436869 0.02857143   0.7645397
bank           19   1.2317460 0.02564103   0.7605182
age            21   6.5151515 0.02941176   0.7562870
```

Lastly degree:

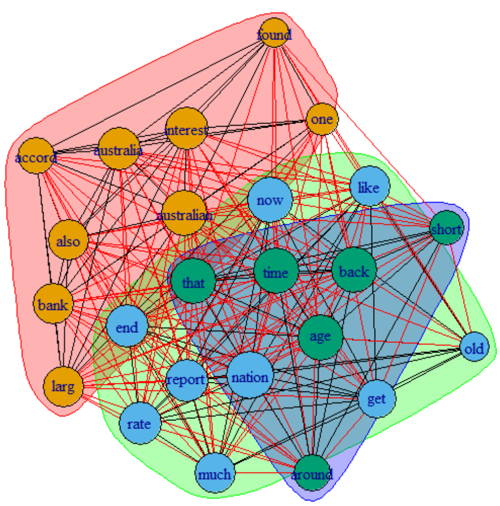
```
> head(statsT[order(-statsT$degree),])
           degree betweenness  closeness eigenvector
nation         22   4.5436869 0.02857143   0.7645397
now            21  11.7051587 0.03125000   0.5895256
australian     21   0.1428571 0.02325581   1.0000000
that           21  11.8615079 0.03225806   0.5717414
time           21   1.3289683 0.02439024   0.8168008
age            21   6.5151515 0.02941176   0.7562870
```

As observed from the data, the node 'nation' has the highest degree (22), indicating that it is connected to the greatest number of edges, making it one of the most important nodes. Additionally, the node 'australian' has the highest Eigenvector centrality (1), suggesting its significance as one of the most important nodes. Furthermore, the node 'now' exhibits the second-highest values in terms of closeness (0.03125000), betweenness (11.705159), and degree (21), solidifying its position as one of the most important nodes. Lastly, the node 'that' possesses the highest betweenness (11.861508) and closeness (0.03225806) measures, further emphasizing its importance as one of the most crucial nodes. In conclusion, the most important nodes in the network are 'nation', 'australian', 'now' and 'that'.

Now I have highlighted the most important nodes in the graph using different colors. Health2.txt is shown in red, Business4.txt in yellow, and Science4.txt in green. Additionally, I have adjusted the size of the nodes based on their degrees, so nodes with higher degrees appear larger:

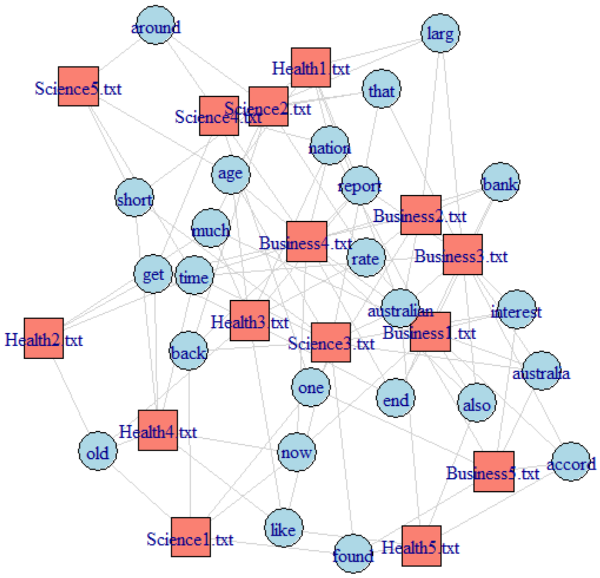


Also use the community detection for this part is a good way to see the relationship between nodes:



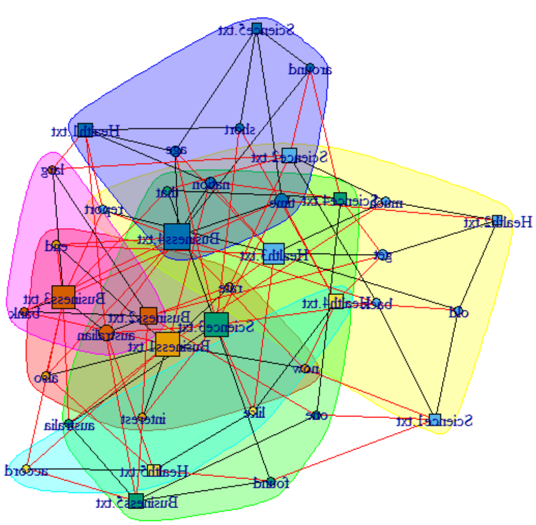
We can see for token we have 3 different part which in each part node will highly be connected and the node have higher degree will be bigger in this network.

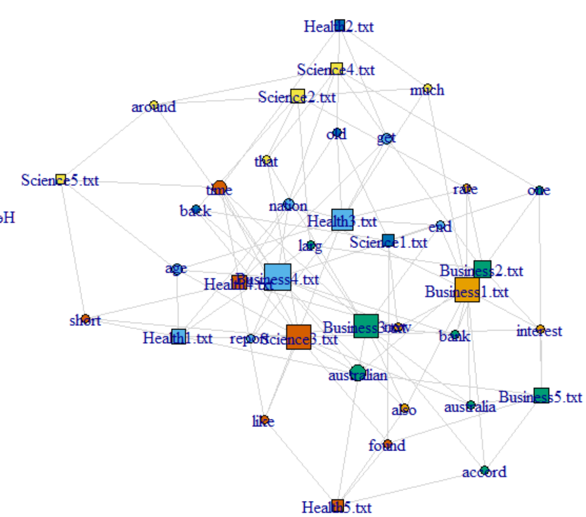
(7). In this part, we create a bipartite (two-mode) network of our corpus. There are two type of nodes which is document ID and tokens. I use the code from lecture node to transform my data into suitable format.



From above network we can see there are two types of nodes, one is light blue circle, which is for tokens. Other one is salmon color square which shows the document ID/document name. It is hard to see the relationship between each node and there are not any clear groups.

So, we need to do some improvement, so we use what we do in question 5 which are spinglass algorithm and cluster_fast_greedy algorithm, they are both community detection method to make a more useful graph. The graph below uses the degree of each node to change the size of the nodes and divide them into different groups. cluster_fast_greedy algorithm divides them into 5 groups and use different color as background, but it has too many groups which make people hard to see the things. So, I mainly look at the spinglass algorithm, this one is much better in here we can see the Business is bigger one here, which means it connect to most of node, it is a very important node. Also, it can show more information in the network, for example, Business3.txt, Business2.txt and Business5.txt are in the same group, they all have words 'australia', 'australian', 'accord', 'lang', 'old', 'bank' and 'one'. The word 'australian' appear most, the





Appendix:

	accord	also	australia	bank	end	interest	much	nation	now	rate	australiar	get	larg	one	report	that	time	age	around	back	found	short	old	like
Business1	1	2	1	1	1	3	1	1	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Business2	0	0	0	3	0	2	0	0	0	1	2	1	1	1	0	0	0	0	0	0	0	0	0	0
Business3	1	1	2	3	1	0	0	0	0	0	1	0	1	0	2	1	1	0	0	0	0	0	0	0
Business4	0	1	0	1	1	0	0	2	1	0	2	0	0	0	1	0	1	5	1	1	0	0	0	0
Business5	1	0	2	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0
Health1.t	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	0	0	1	0	0	0	1	0	0
Health2.t	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	2	0
Health3.t	0	0	0	0	1	0	1	1	0	2	0	2	0	0	2	0	0	1	0	0	0	0	2	1
Health4.t	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	1	2
Health5.t	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	3
Science1.t	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	0	2	0
Science2.t	0	0	0	0	0	0	2	0	0	1	0	0	1	0	0	1	0	0	1	1	0	0	0	0
Science3.t	0	0	1	0	0	1	0	0	0	0	2	0	0	0	0	1	1	1	0	1	2	1	0	1
Science4.t	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	1	0	0	0	0	0
Science5.t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	0	0	1	0	0

R Script:

```
#Task1, I select the text from ABC News

################################################################################

#Task2

setwd("C:/Users/WENGCHENLONGJIE/Desktop/FIT3152/Assignment3")

#first import all the package I need for this question

rm(list = ls()) #clean up the workplace at start

library(slam)

library(tm)

library(SnowballC)


#then read the folder which contain all the text file we needed

cname = file.path(".", "text_folder") #

docs = Corpus(DirSource((cname))) #add all the file in to Corpus

print(summary(docs)) #print all the file we have here



################################################################################
```

```
#Task3

#the main parts going with these step

#Tokenise, Convert case, Filtering – including removing stop words, Stemming

#Tokenise change the word to token

toSpace = content_transformer(function(x,pattern) gsub(pattern, " ", x))

docs = tm_map(docs, toSpace, "-") #remove'-'

docs = tm_map(docs, toSpace, "–") #remove'-'

docs = tm_map(docs, toSpace, "—") #remove'-'

docs = tm_map(docs, removeNumbers) # remove numbers

docs = tm_map(docs, removePunctuation) #remove punctuation

docs = tm_map(docs, content_transformer(tolower)) #change words to lower case


#Filter words

# Remove stop words and white space

docs = tm_map(docs, removeWords, stopwords("english"))

docs = tm_map(docs, stripWhitespace)


#stem

docs = tm_map(docs, stemDocument, language = "english") #change the word with
similar meaning to same token


#create document term matrix

dtm = DocumentTermMatrix(docs)


#check word frequencies

freq = colSums(as.matrix(dtm))  #count same token

length(freq)  #how many token in total
```

```
ord = order(freq) #frequency in order

freq[head(ord)] #less

freq[tail(ord)] #most


#remove the word whihc might appear in every document

docs = tm_map(docs, toSpace, "will")

docs = tm_map(docs, toSpace, "year")

docs = tm_map(docs, toSpace, "said")

docs = tm_map(docs, stripWhitespace)


#create document term matrix

dtm = DocumentTermMatrix(docs)


#check word frequencies

freq = colSums(as.matrix(dtm))  #count same token

length(freq)  #how many token in total

ord = order(freq) #frequency in order

freq[head(ord)] #less

freq[tail(ord)] #most


#frequency of frequencies

head(table(freq), 10) #make frequency table which appear 1-10 times

tail(table(freq), 10)

dim(dtm) #show the number of files and number of different token

dtms = removeSparseTerms(dtm, 0.75) #here we want only left about 20 tokens

dim(dtms) #to show how many token after we doing it

#to show the dtms details
```

```
inspect(dtms)

dtms = as.matrix(dtms)

write.csv(dtms, "dtms.csv")



################################################################################
####

#Task4

#this is the Euclidean distance way to do it.

distmatrix = dist(scale(dtms))

fit = hclust(distmatrix, method = "ward.D")

tree = as.dendrogram(fit)

tree = color_branches(tree, k = 3) #Color the clusters of clustering results

plot(tree, main = 'Euclidean distance')


#this is the Cosine Distance way to do it.

library(proxy)

#install.packages("dendextend")

library(dendextend)

dtm_cos = proxy::dist(as.matrix(dtms), method = "cosine")

fit_cos = hclust(dtm_cos)

tree_cos = as.dendrogram(fit_cos) #Convert fit into a visual tree diagram

tree_cos = color_branches(tree_cos, k = 3) #Color the clusters of clustering results

plot(tree_cos, main='Cosine Distance')


cutfit = cutree(fit, k = 3)

cutfit #use to get more information about Which cluster is each sample assigned to
```

```
sort(cutfit)


cutfit_cos = cutree(fit_cos, k = 3)

cutfit_cos #use to get more information about Which cluster is each sample assigned to

sort(cutfit_cos) # Output the cluster to which each document belongs


library(fpc)

#install.packages("fpc")

#Silhouette Coefficient

silhouette_score <- cluster.stats(distmatrix, cutfit)$avg.silwidth #should from -1 to 1,
higher value means better

silhouette_score #0.06029111

silhouette_score_cos <- cluster.stats(distmatrix, cutfit_cos)$avg.silwidth

silhouette_score_cos #-0.03370824


#Calinski-Harabasz Index

ch_index <- cluster.stats(distmatrix, cutfit)$ch #higher is better

ch_index #2.180116

ch_index_cos <- cluster.stats(distmatrix, cutfit_cos)$ch

ch_index_cos #1.143047



##########################################################################
####

#Task5

#first import package we need in this part

library(igraph)

library(igraphdata)
```

```
#these is the steps to make network

# start with original document-term matrix

dtmsx = as.matrix(dtms)

# convert to binary matrix

dtmsx = as.matrix((dtmsx > 0) + 0)

# multiply binary matrix by its transpose

ByAbsMatrix = dtmsx %*% t(dtmsx)

# make leading diagonal zero

diag(ByAbsMatrix) = 0

# create graph object

ByAbs = graph_from_adjacency_matrix(ByAbsMatrix,mode = "undirected", weighted =
TRUE)

set.seed(29334152)

plot(ByAbs)


#now start to find the most important document in the network

d = as.table(degree(ByAbs)) #the degree of each document

b = as.table(betweenness(ByAbs)) #the betweenness of each document

c = as.table(closeness(ByAbs)) #the closeness of each document

e = as.table(evcent(ByAbs)$vector) #the eigenvector of each document

stats = as.data.frame(rbind(d,b,c,e))#combine the d,b,c,e together

stats = as.data.frame(t(stats)) #change col to row

colnames(stats) = c("degree", "betweenness", "closeness", "eigenvector") #assign
correct name

stats

#sort and explore key nodes
```

```
head(stats[order(-stats$betweenness),]) #we only need to know the high most so use head

head(stats[order(-stats$closeness),])

head(stats[order(-stats$eigenvector),])

head(stats[order(-stats$degree),])


V(ByAbs)['Business4.txt']$color = "yellow"

V(ByAbs)['Health2.txt']$color = "red"

V(ByAbs)['Science4.txt']$color = "green"

set.seed(29334152)

plot(ByAbs, vertex.size = degree(ByAbs))


communities <- cluster_fast_greedy(ByAbs)  #community detection

plot(communities, ByAbs, vertex.color = communities$membership, vertex.size = degree(ByAbs))


spectral <- cluster_spinglass(ByAbs) #community detection

membership <- membership(spectral)

plot(ByAbs, vertex.color = membership, vertex.label = V(ByAbs)$name, vertex.size = degree(ByAbs))




##############################################################################

#Task6

#these is the steps to make network

# start with original document-term matrix
```

```
dtmsx = as.matrix(dtms)

# convert to binary matrix

dtmsx = as.matrix((dtmsx > 0) + 0)

# multiply binary matrix by its transpose

ByTokenMatrix = t(dtmsx) %*% dtmsx

# make leading diagonal zero

diag(ByTokenMatrix) = 0

# create graph object

ByToken = graph_from_adjacency_matrix(ByTokenMatrix,mode = "undirected",
weighted = TRUE)

set.seed(29334152)

plot(ByToken)


#now start to find the most important document in the network

dT = as.table(degree(ByToken)) #the degree of each document

bT = as.table(betweenness(ByToken)) #the betweenness of each document

cT = as.table(closeness(ByToken)) #the closeness of each document

eT = as.table(evcent(ByToken)$vector) #the eigenvector of each document

statsT = as.data.frame(rbind(dT,bT,cT,eT))#combine the dT,bT,cT,eT together

statsT = as.data.frame(t(statsT)) #change col to row

colnames(statsT) = c("degree", "betweenness", "closeness", "eigenvector") #assign
correct name

statsT

#sort and explore key nodes

head(statsT[order(-statsT$betweenness),]) #we only need to know the high most so use
head

head(statsT[order(-statsT$closeness),])

head(statsT[order(-statsT$eigenvector),])
```

```
head(statsT[order(-statsT$degree),])

#highlight the most important point
V(ByToken)['nation']$color = "yellow"
V(ByToken)['australian']$color = "red"
V(ByToken)['now']$color = "green"
V(ByToken)['that']$color = "blue"
set.seed(29334152)
plot(ByToken, vertex.size = degree(ByToken))

communities <- cluster_fast_greedy(ByToken)  #community detection
plot(communities, ByToken, vertex.color = communities$membership, vertex.size =
degree(ByToken))




############################################################################
####
#Task7

# start with document term matrix dtms
dtmsa = as.data.frame(dtms) # clone dtms
dtmsa$ABS = rownames(dtmsa) # add row names
dtmsb = data.frame()
for (i in 1:nrow(dtmsa)){
  for (j in 1:(ncol(dtmsa)-1)){
    touse = cbind(dtmsa[i,j], dtmsa[i,ncol(dtmsa)],
                colnames(dtmsa[j]))
```

```
    dtmsb = rbind(dtmsb, touse ) } } # close loops
colnames(dtmsb) = c("weight", "abs", "token")
dtmsc = dtmsb[dtmsb$weight != 0,] # delete 0 weights

# put colunms in order: abs, token, weight
dtmsc = dtmsc[,c(2,3,1)]

# create graph object and declare bipartite
g <- graph.data.frame(dtmsc, directed=FALSE)
bipartite.mapping(g)
V(g)$type <- bipartite_mapping(g)$type
V(g)$color <- ifelse(V(g)$type, "lightblue", "salmon")
V(g)$shape <- ifelse(V(g)$type, "circle", "square")
E(g)$color <- "lightgray"
set.seed(29334152)
plot(g)

communities <- cluster_fast_greedy(g)  #community detection
plot(communities, g, vertex.color = communities$membership, vertex.size = degree(g))

spectral <- cluster_spinglass(g) #community detection
membership <- membership(spectral)
plot(g, vertex.color = membership, vertex.label = V(g)$name, vertex.size = degree(g))
```

Reference:

Judd, B. (2023, June 7). The first 'virgin birth' in crocodiles has been discovered, but what triggers some animals to go it alone? Retrieved from https://www.abc.net.au/news/science/2023-06-07/parthenogenesis-virgin-birth-american-crocodile/102440852

Kilvert, N. (2023, June 6). Carbon dioxide in atmosphere hits new record high as scientists call for 'every effort to slash carbon pollution'. Retrieved from https://www.abc.net.au/news/science/2023-06-06/science-atmospheric-co2-hits-record-high/102444412

Francis, A. (2023, June 6). Dog fence affects growth rate of young red kangaroos exposed to dingoes, research finds. Retrieved from https://www.abc.net.au/news/2023-06-06/australian-dog-fence-affects-kangaroo-growth-rate/102426930

Maguire, D. (2023, June 3). NASA is sending a 'message in a bottle' towards Jupiter. Here's how you can add your name to it. Retrieved from https://www.abc.net.au/news/2023-06-03/nasa-sends-message-in-a-bottle-to-jupiter-europa-clipper/102436722

Kean, Z. (2023, June 5). The last King Island emu died a stranger in a foreign land. Retrieved from https://www.abc.net.au/news/2023-06-05/the-king-island-emu-died-alone-in-paris/102425608

Morse, D. (2023, June 6). National audit office releases scathing report of Morrison-government health funding program. Retrieved from https://www.abc.net.au/news/2023-06-06/scathing-report-into-morrison-government-health-funding-program/102443402

Clayton, R., & Brown, M. (2023, June 6). Older brothers and sisters could help build babies' defence against allergies, findings suggest. Retrieved from https://www.abc.net.au/news/2023-06-06/food-allergies-gut-bacteria-siblings-protect-babies-study/102440522

Thorne, L. (2023, June 6). Flu season has started early, and doctors hold fears for children amid low vaccination rates. Retrieved from https://www.abc.net.au/news/2023-06-06/influenza-flu-season-affecting-kids-children-hospital-cases-up/102387982

Smoleniec, B. (2023, June 6). A lack of affordable dental care means Sherry has lost several teeth — and her confidence. Retrieved from https://www.abc.net.au/news/2023-06-06/lack-of-affordable-dental-care-means-sherry-lost-several-teeth/102442812

Voloder, D. (2023, June 7). Culturally diverse women share their stories of complicated pregnancies and birth interventions. Retrieved from https://www.abc.net.au/news/2023-06-07/migrant-women-more-at-risk-of-pregnancy-complications/102415338

Hutchens, G., & Pupazzoni, R. (2023, June 7). Australia's economy grew by 0.2pc in March quarter, signalling marked slowdown in GDP. Retrieved from https://www.abc.net.au/news/2023-06-07/gdp-march-quarter-0-2-per-cent-australia/102444792

Janda, M. (2023, June 7). Philip Lowe warns RBA must use 'tool it has' to get inflation under control — and that may mean more rate rises. Retrieved from https://www.abc.net.au/news/2023-06-07/rba-philip-lowe-interest-rates-inflation/102448754

Maguire, D. (2023, June 7). Cheques will be phased out by 2030 as mobile wallet use sky-rockets. Retrieved from https://www.abc.net.au/news/2023-06-07/australian-government-to-phase-out-cheques-by-2030/102449560

Borys, S. (2023, June 7). Aged care task force looking at new taxes, and more self-funding for wealthy elderly Australians. Retrieved from https://www.abc.net.au/news/2023-06-07/minister-leaves-door-open-for-aged-care-levy/102449472

Whitson, R. (2023, June 7). Jenny Craig collapses in Australia and New Zealand after no buyer found for weight-loss company. Retrieved from https://www.abc.net.au/news/2023-06-07/jenny-craig-to-close-australia-new-zealand/102451450