**Experiment Steps**

1. For the complete set of documents, try the functions in lecture 7.

load the raw data.

# > install.packages("tm")

> library(tm)

# > install.packages("textreuse")

> library(textreuse)

# > install.packages("wordcloud")

# > library(wordcloud)

# > install.packages("ggplot2")

# > library("ggplot2");

# > install.packages("wordnet")

# > library("wordnet")

# > install.packages("zipfR")

> library(zipfR)

# > data("acq")

follow lecture 7, use acq[1] as example:

# > SATlow <- tm\_map(acq[1], content\_transformer(tolower))

# 

# > removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]\*", "", x)

# > SATcl <- tm\_map(SATlow, content\_transformer(removeNumPunct))

# > myStopwords <- c(stopwords('english'))

# > SATstop <- tm\_map(SATcl, removeWords, myStopwords)

# 

# > SATtdm2 <-TermDocumentMatrix(SATstop,control=list(wordLengths=c(1,Inf)));

# > freq.terms <- findFreqTerms(SATtdm2,lowfreq = 5);

# 

# > findAssocs(SATtdm2,"states",0.25);

# 

# > term.freq<-rowSums(as.matrix(SATtdm2));

# > term.freq<-subset(term.freq,term.freq>=2);

# > df<-data.frame(term=names(term.freq),freq=term.freq);

# 

# 

# > ggplot(df,aes(x=term,y=freq))+geom\_bar(stat="identity")+xlab("Terms")+ylab("count")+coord\_flip();

# 

# > tdm2<-removeSparseTerms(SATtdm2,sparse=0.50);

# 

# > distMatrix<-dist(scale(tdm2));

# 

# > fit<-hclust(distMatrix,method="ward.D2");

# > plot(fit)

# 

# > rh<-rect.hclust(fit,k=10);

# 

# > tm<-t(distMatrix);

# > k<-6;

# > kr<-kmeans(tm,k);

# > round(kr$centers,digits=3);

# 

# > m1<-as.matrix(tdm2);

# > word.freq<-sort(rowSums(m1),decreasing=T);

# 

# > pal<-brewer.pal(9,"BuGn");

# > pal<-pal[-(1:4)];

# > wordcloud(words=names(word.freq),freq=word.freq,min.freq=2,random.order=F,colors=pal);

# 

b. Find the 10 longest documents (in number of words).

> count=array(dim = 50)

#compute number of words of each docs

> for(i in 1:50){

count[i]=length(tokenize\_words(acq[[i]]$content))

}



#sort the length of these docs and find 10 longest docs

> countindex=array(dim = 10)

> for(i in 1:10){

countindex[i]=which(count==sort(count)[length(count)-i+1])

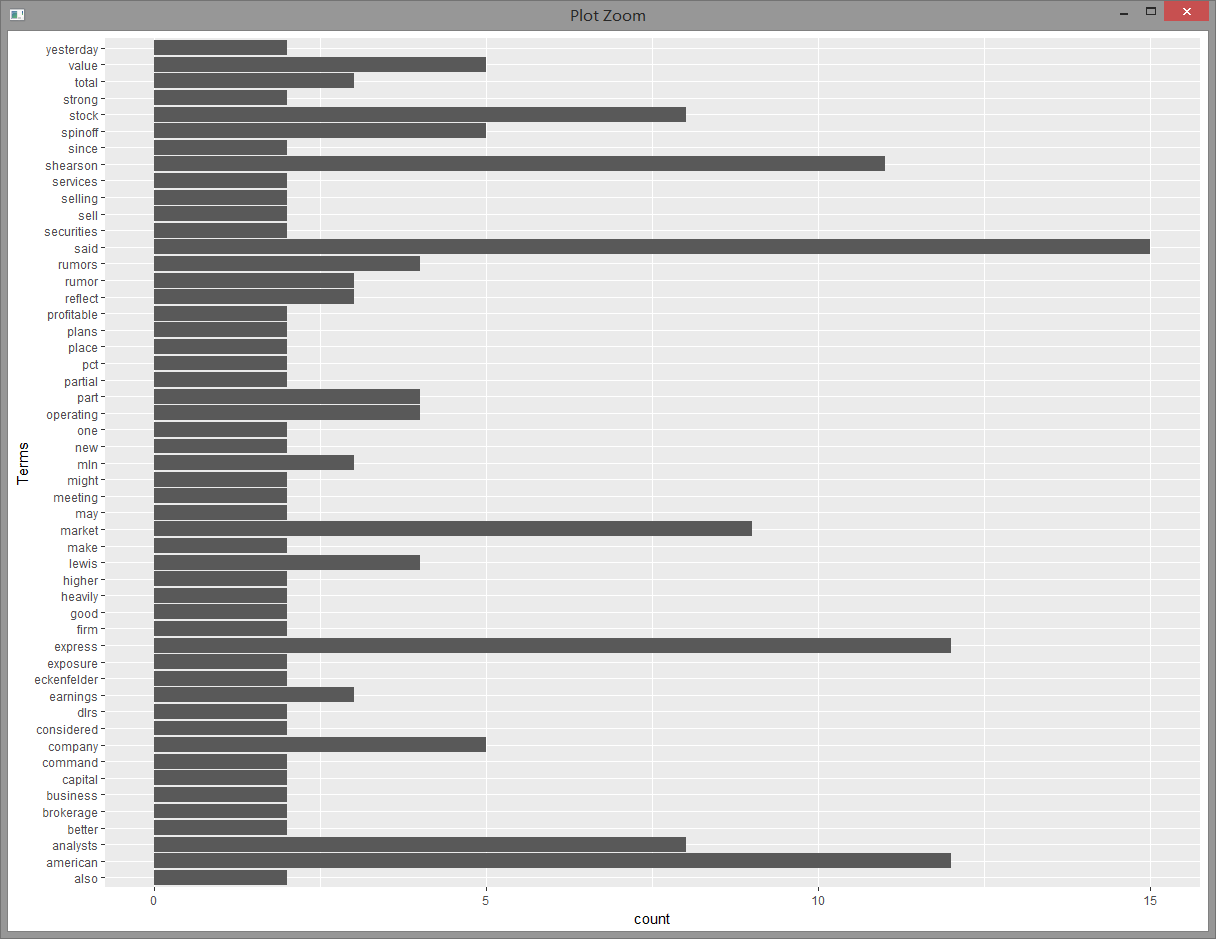
}

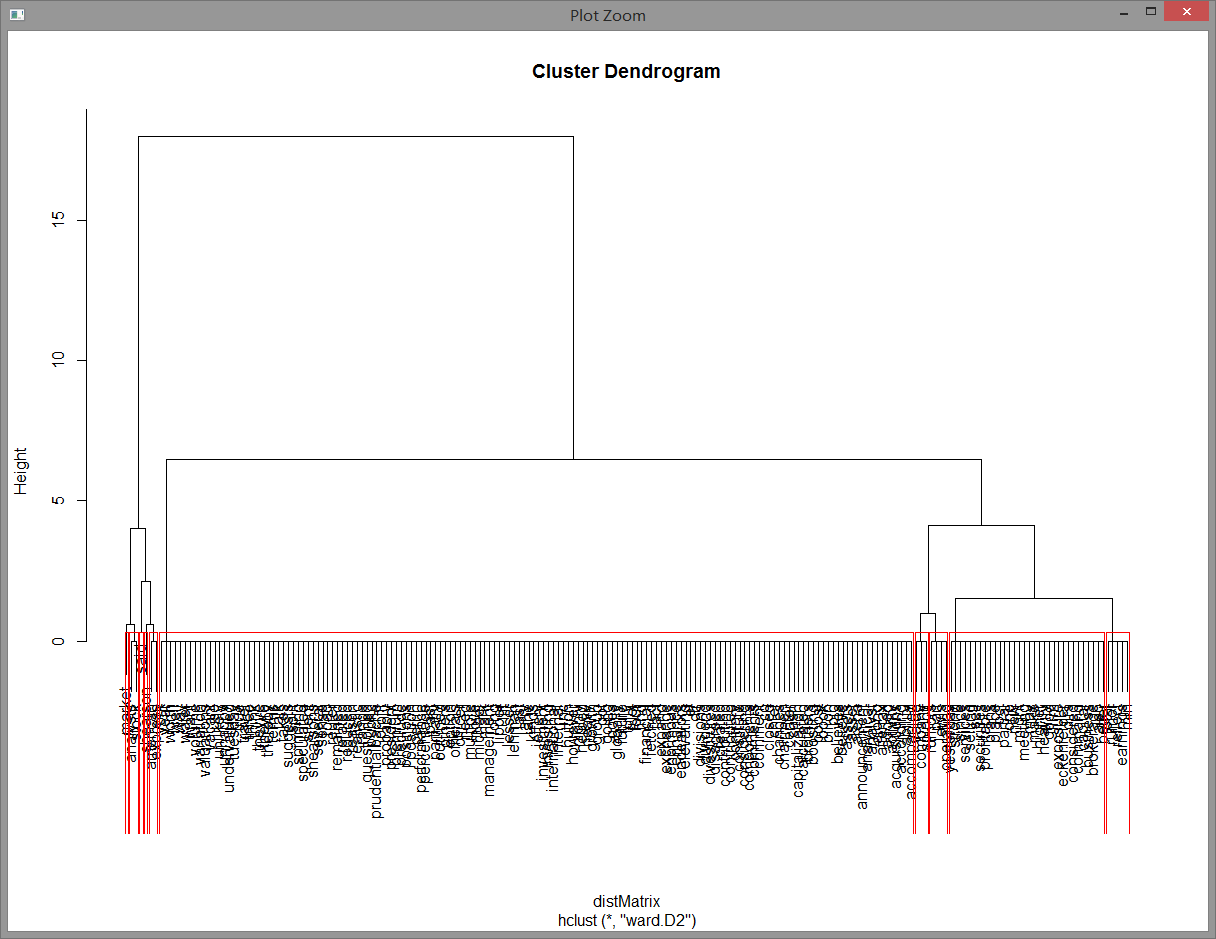


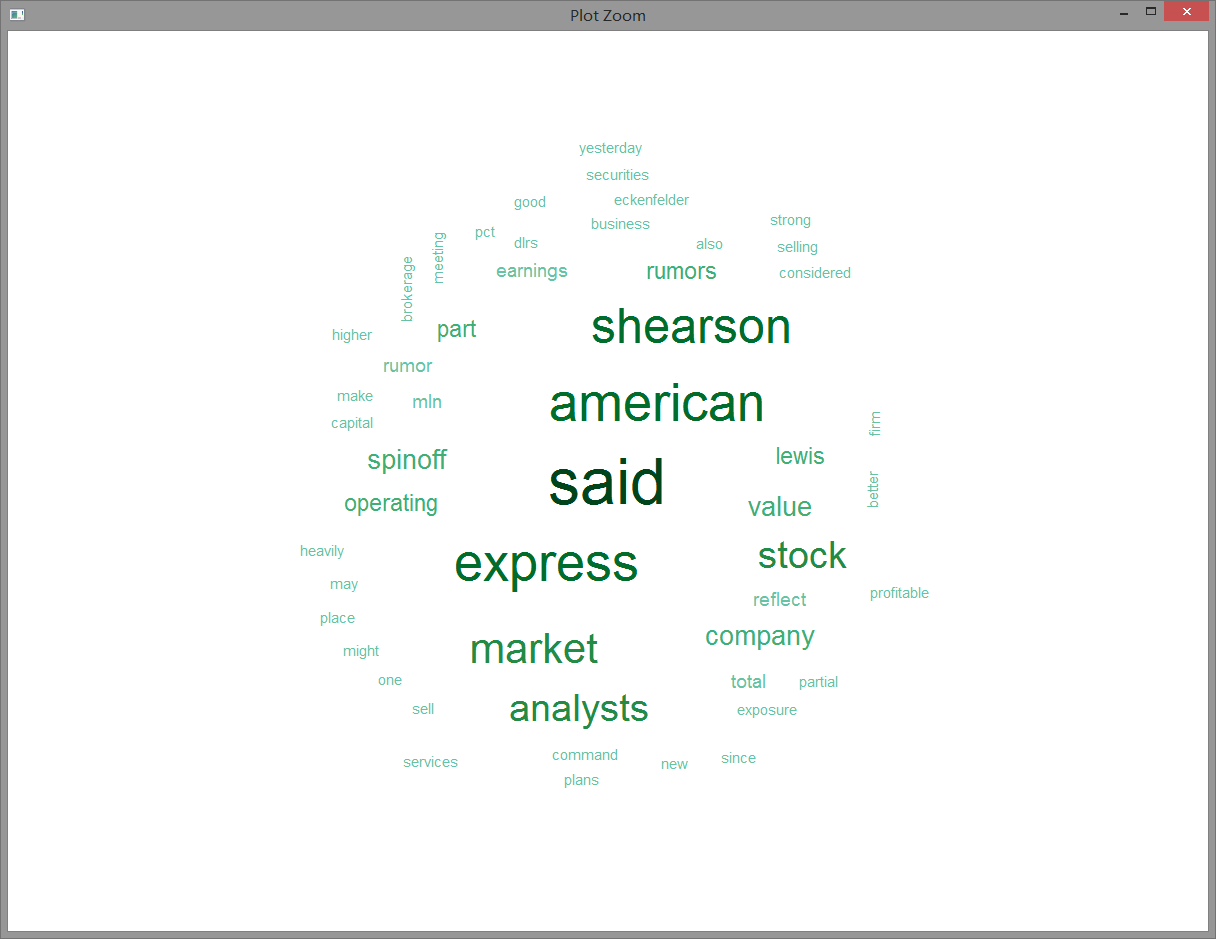
The 10 longest documents are acq[7], acq[25], acq[29], acq[47], acq[19], acq[4], acq[22], acq[42], acq[34], acq[1].

c. For each document work through the examples given in Lecture 7 to display the dendrogram and the WordCloud.

1. acq[7]

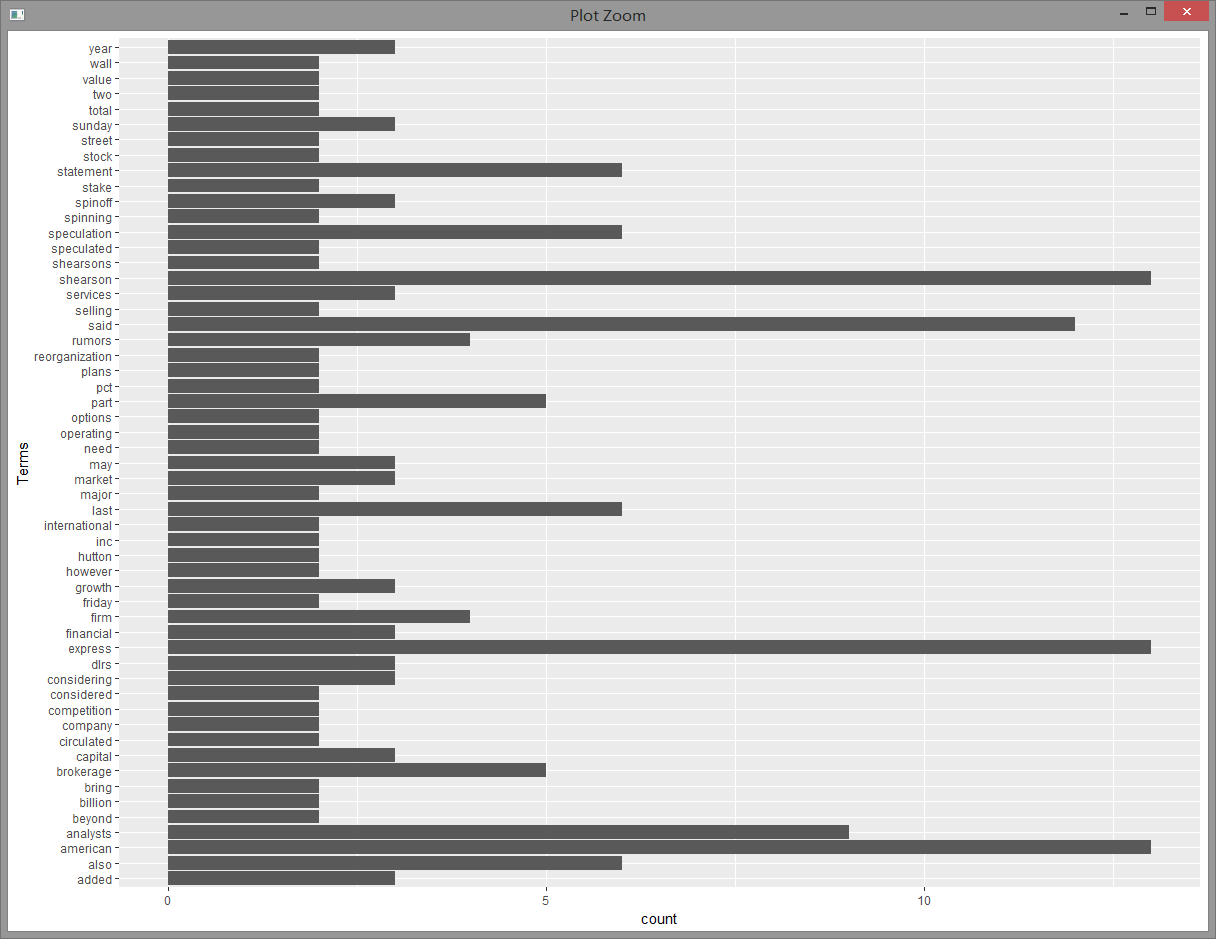
Word frequency:

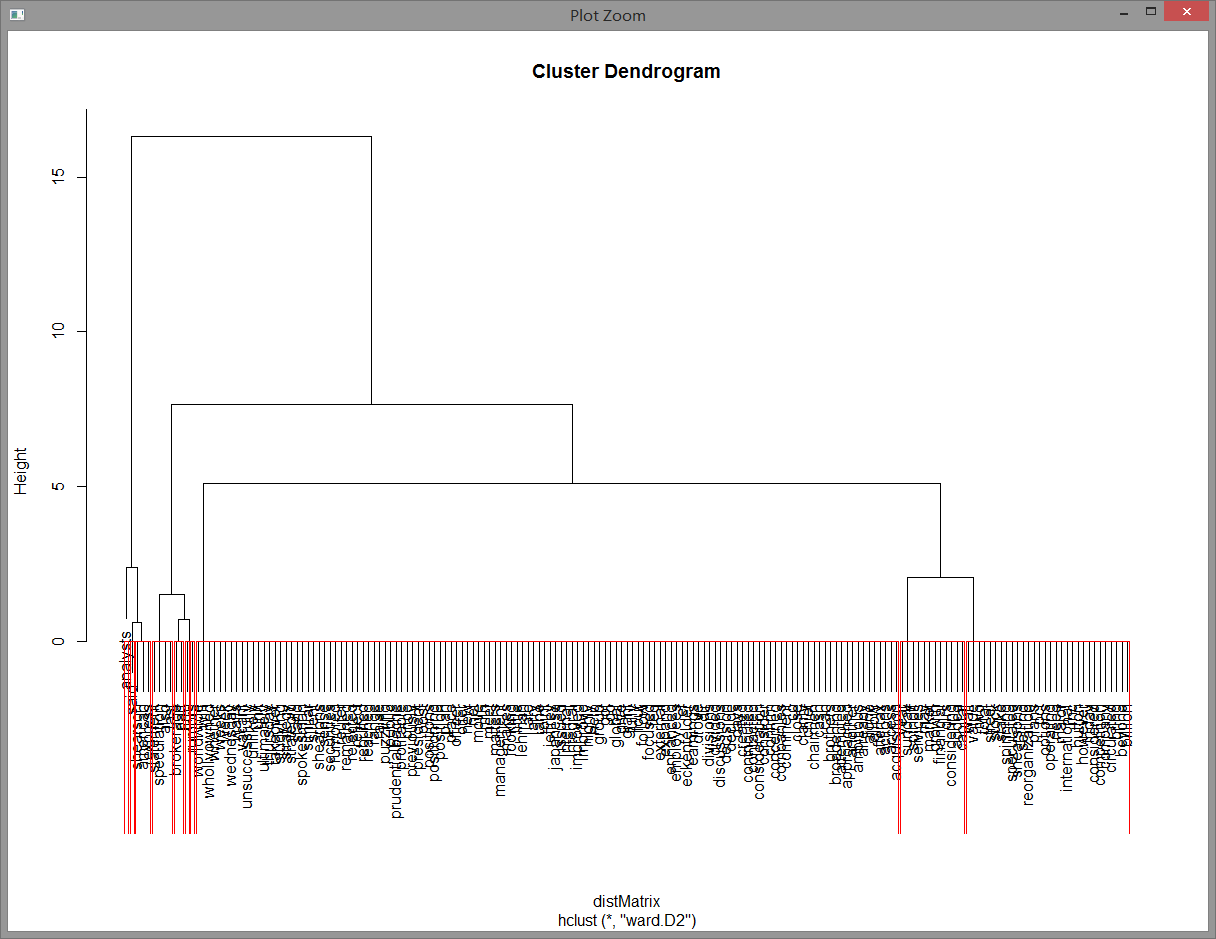
Dendrogram:

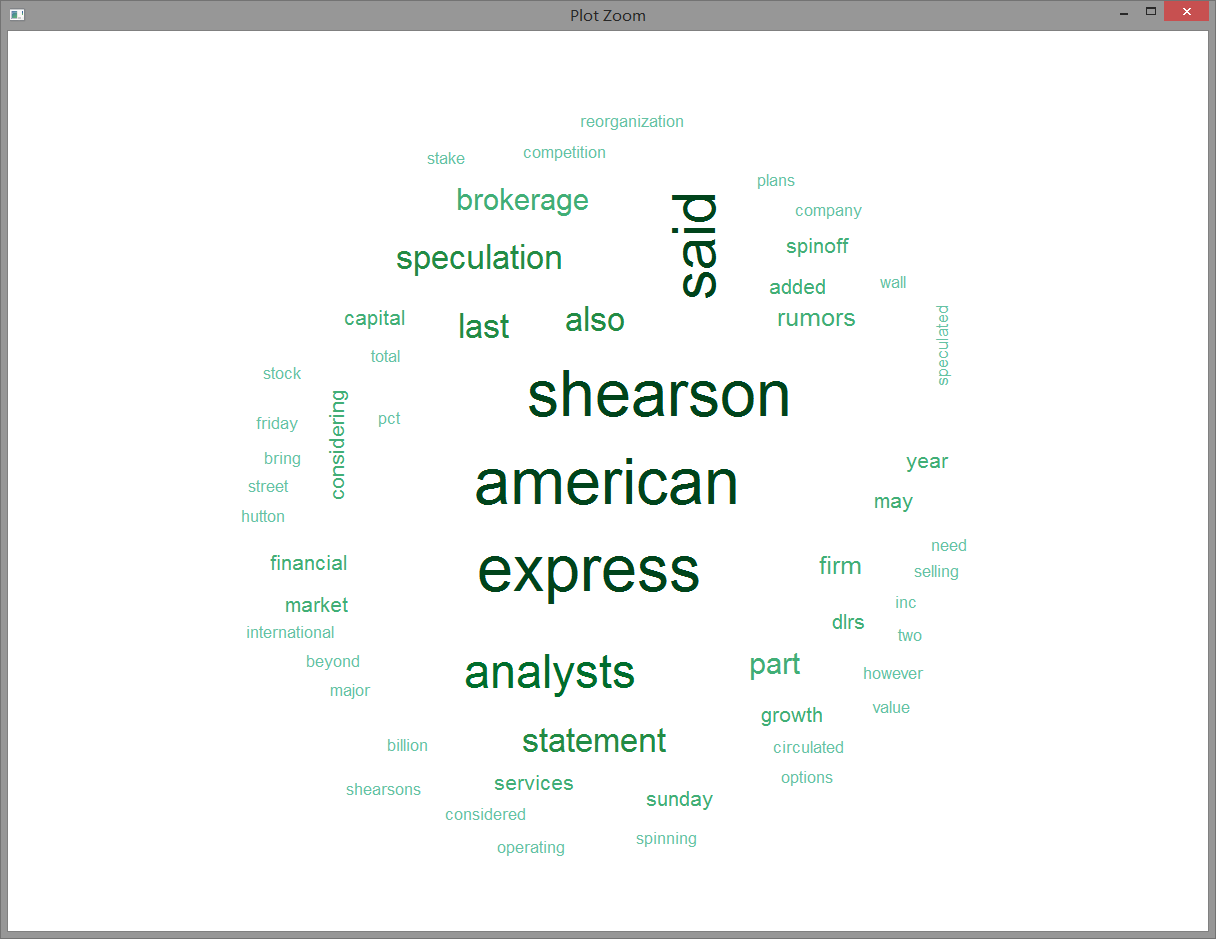
WordCloud:

**Discussion**: “Said” is the most frequently used terms in this document.

**2. acq[25]**

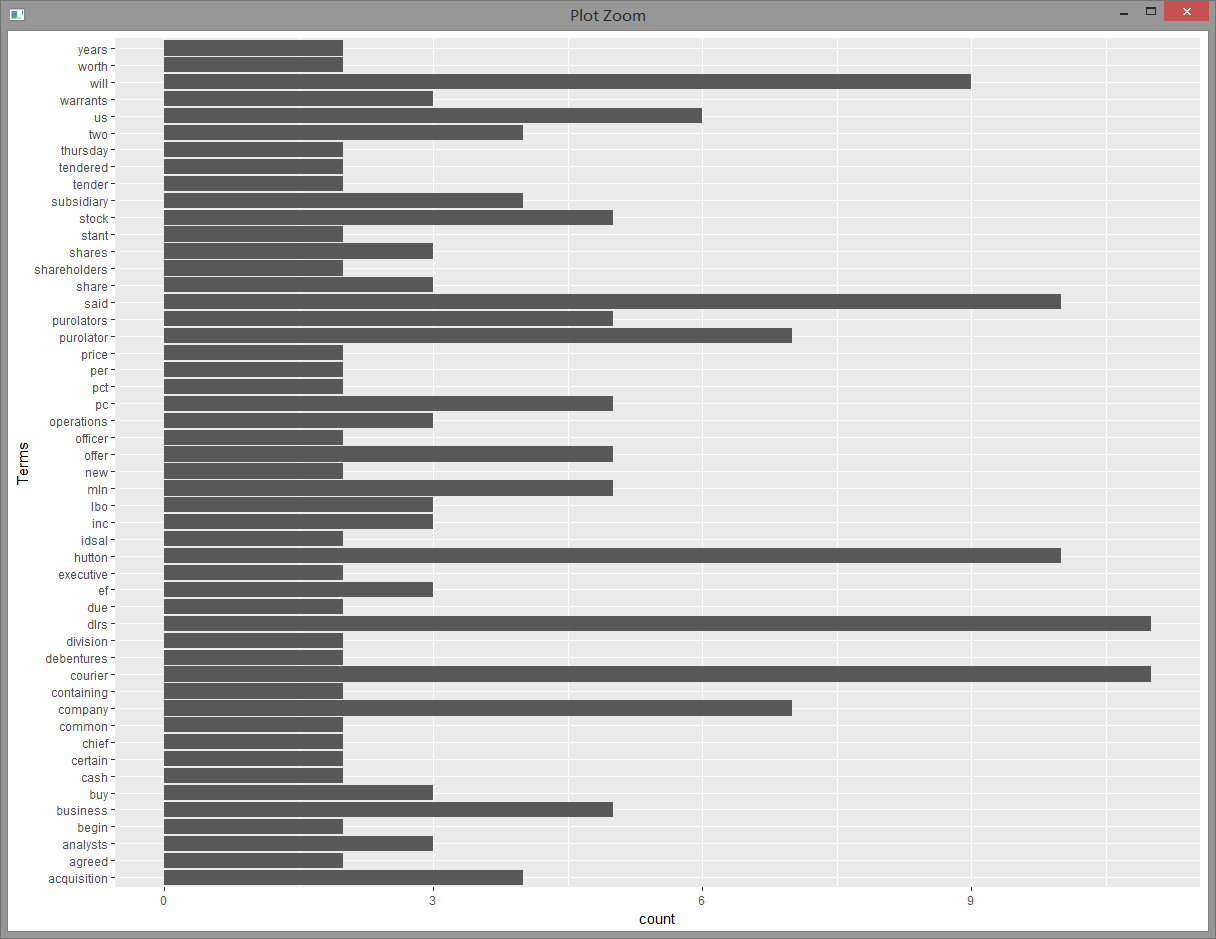
Word frequency:

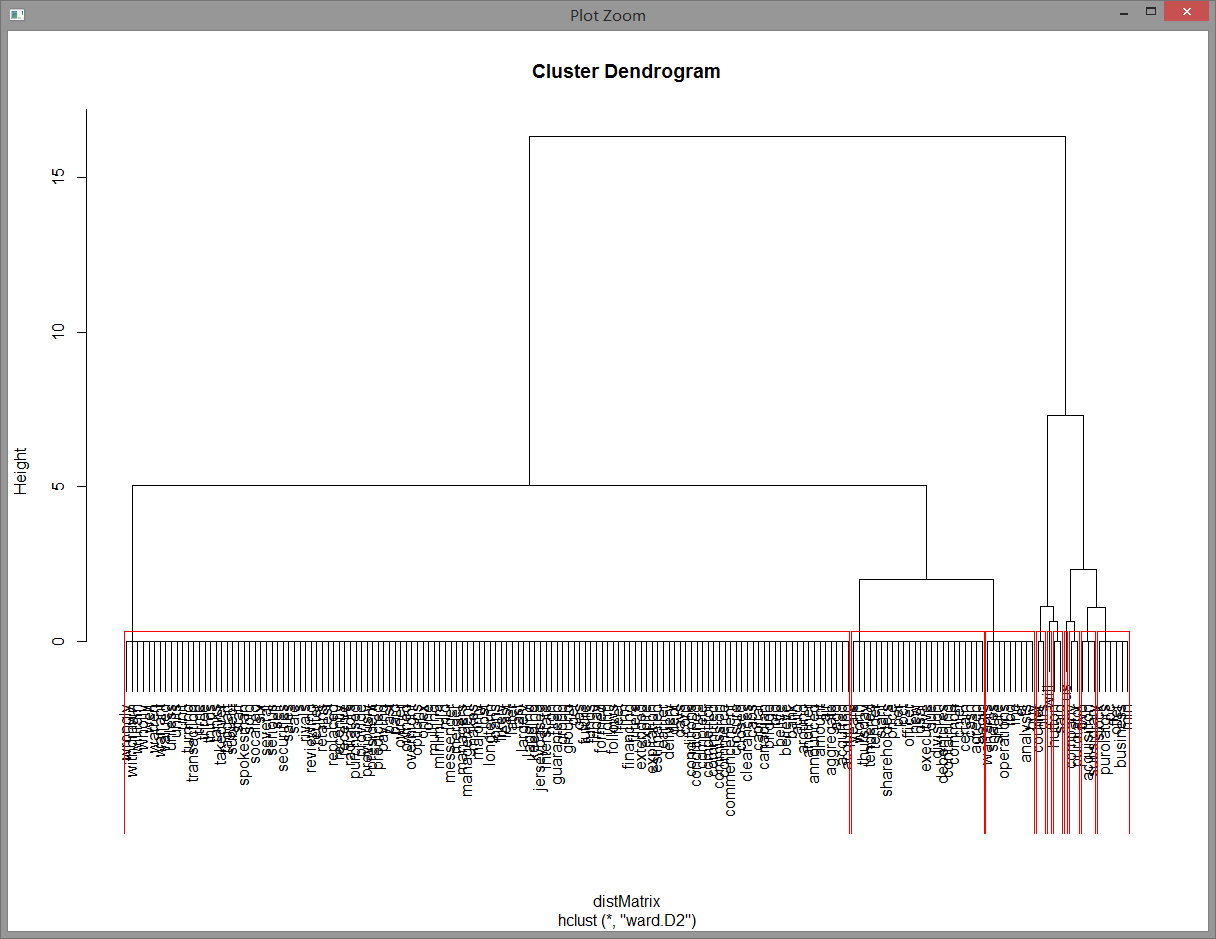
Dendrogram:

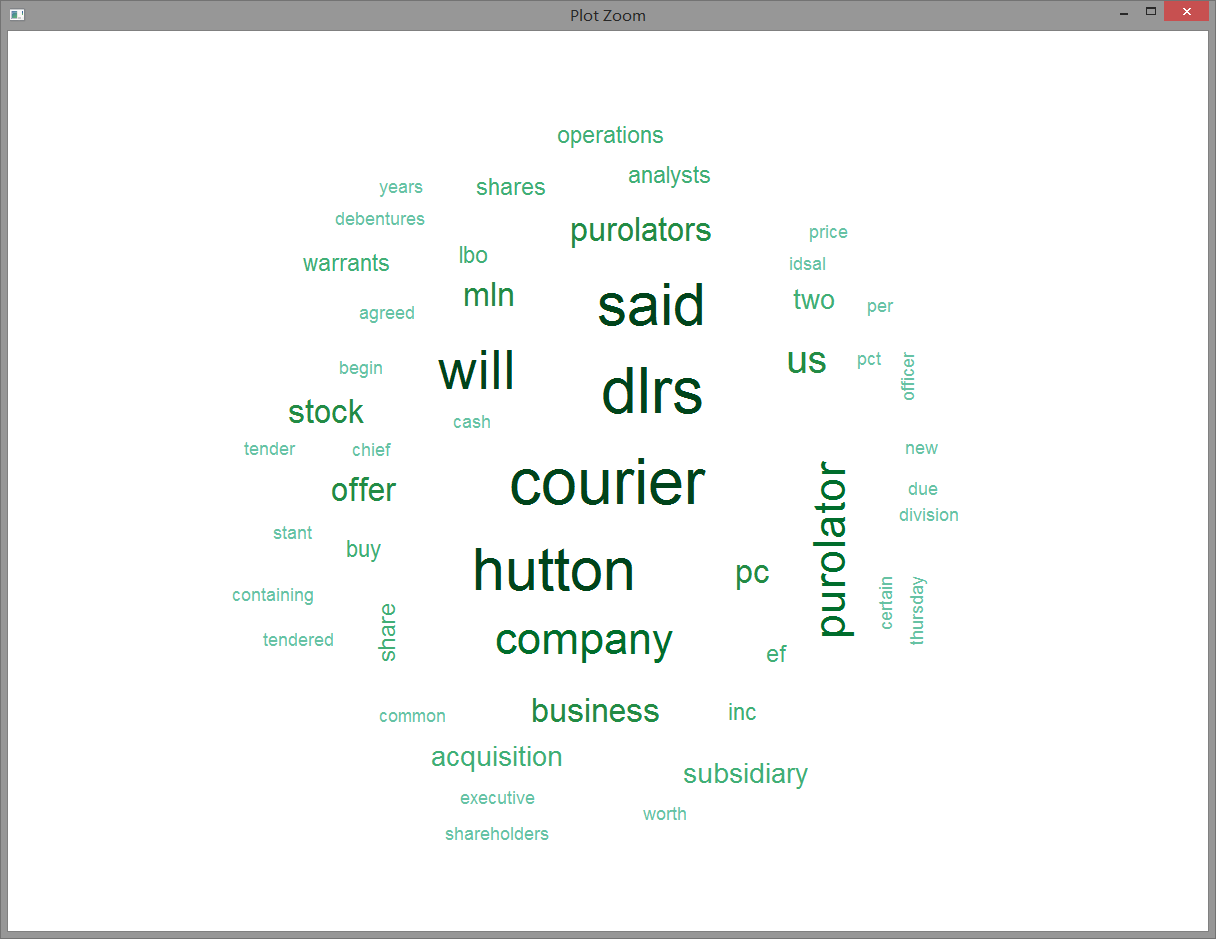
WordCloud:

**Discussion**: “shearson”, “american” and “express” are the most frequently used terms in this document.

**3. acq[29]**

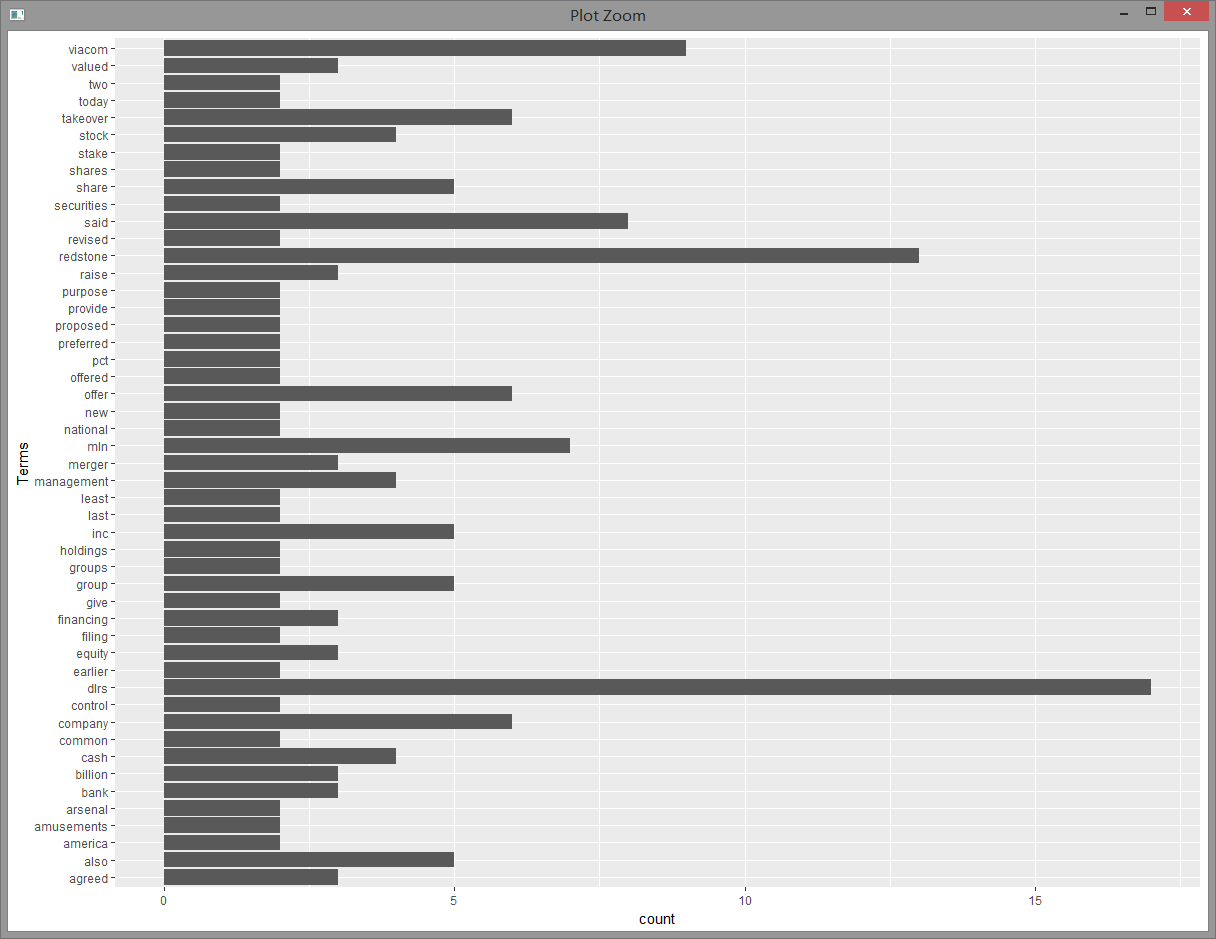
Word frequency:

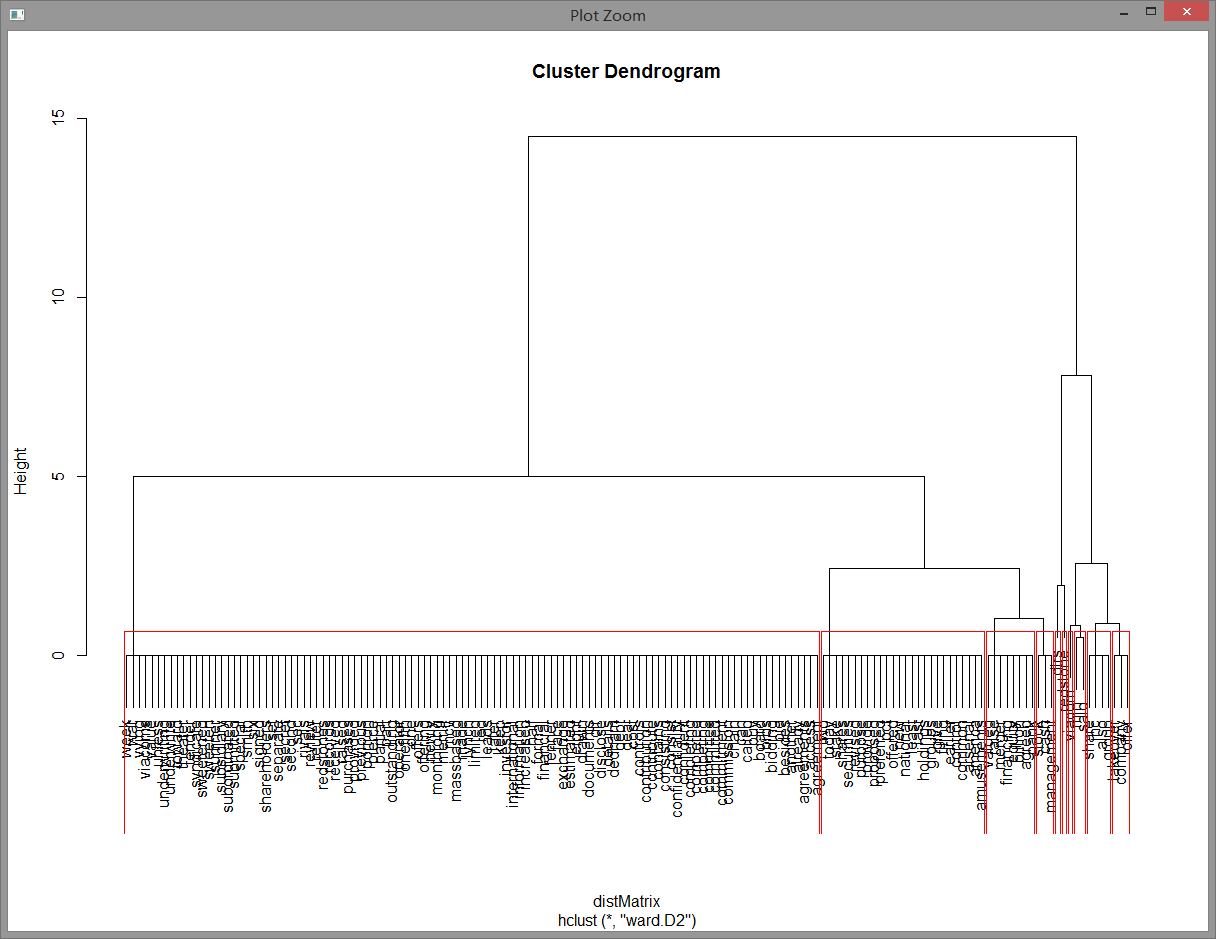
Dendrogram:

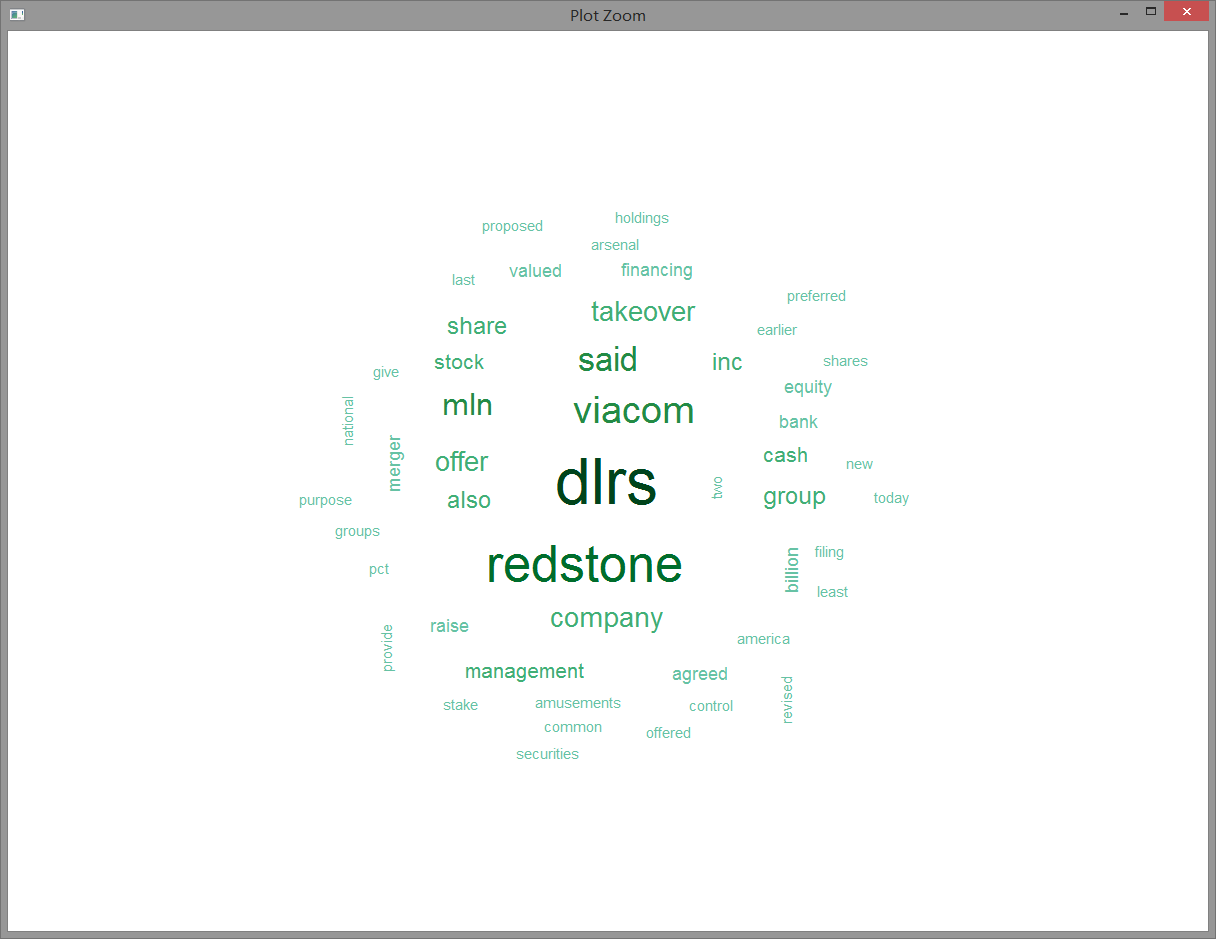
WordCloud:

**Discussion**: “Said”, “dlrs”, “courier” and “hutton” are the most frequently used terms in this document.

**4. acq[47]**

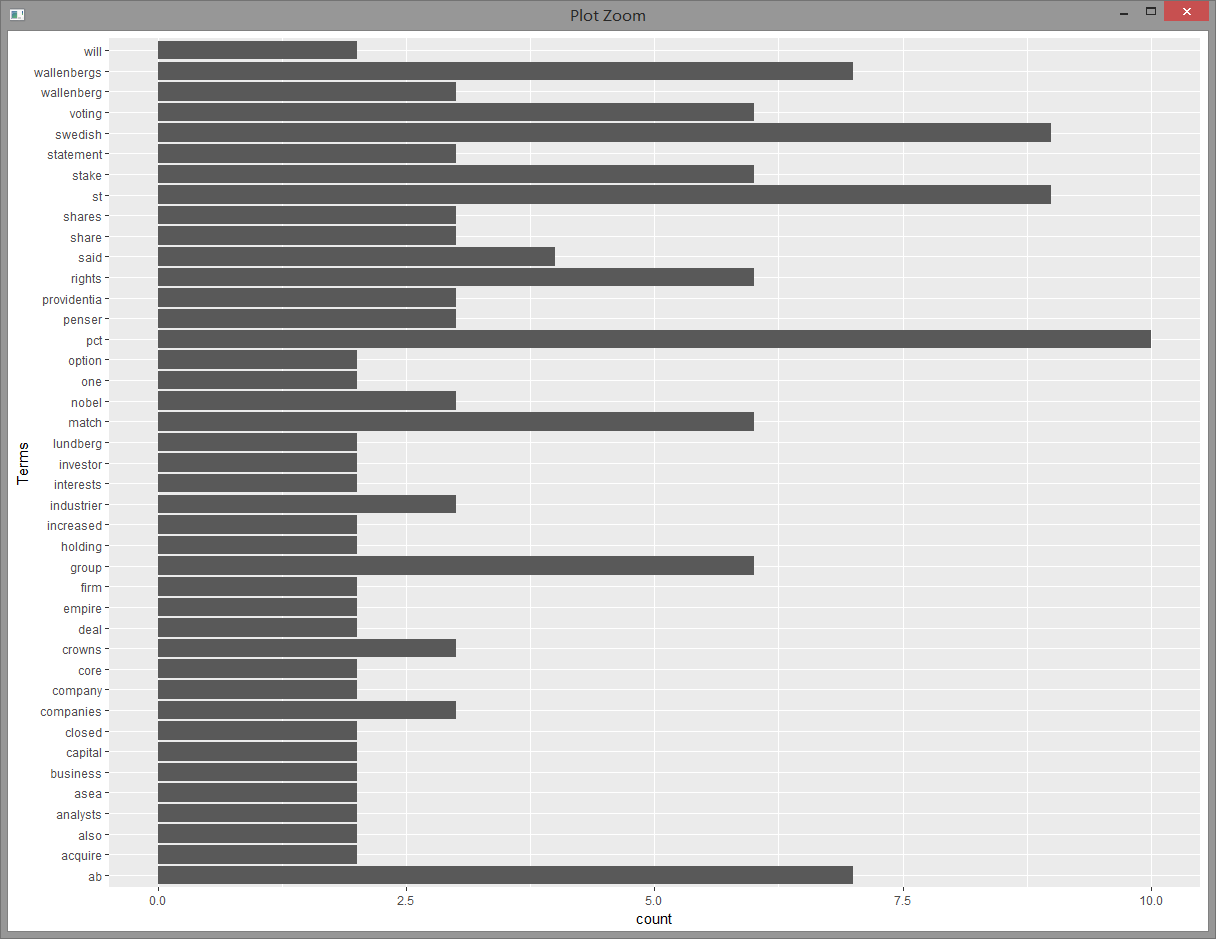
Word frequency:

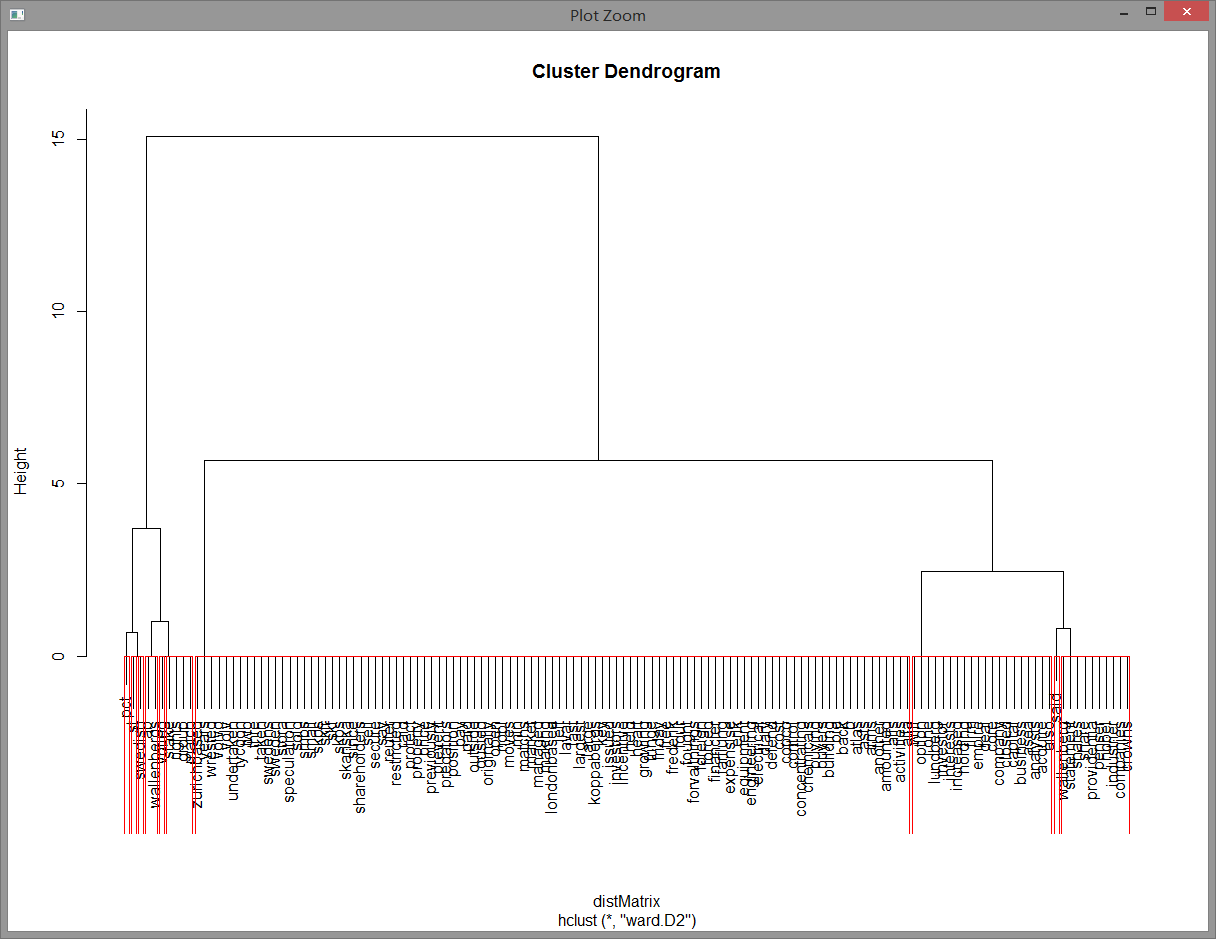
Dendrogram:

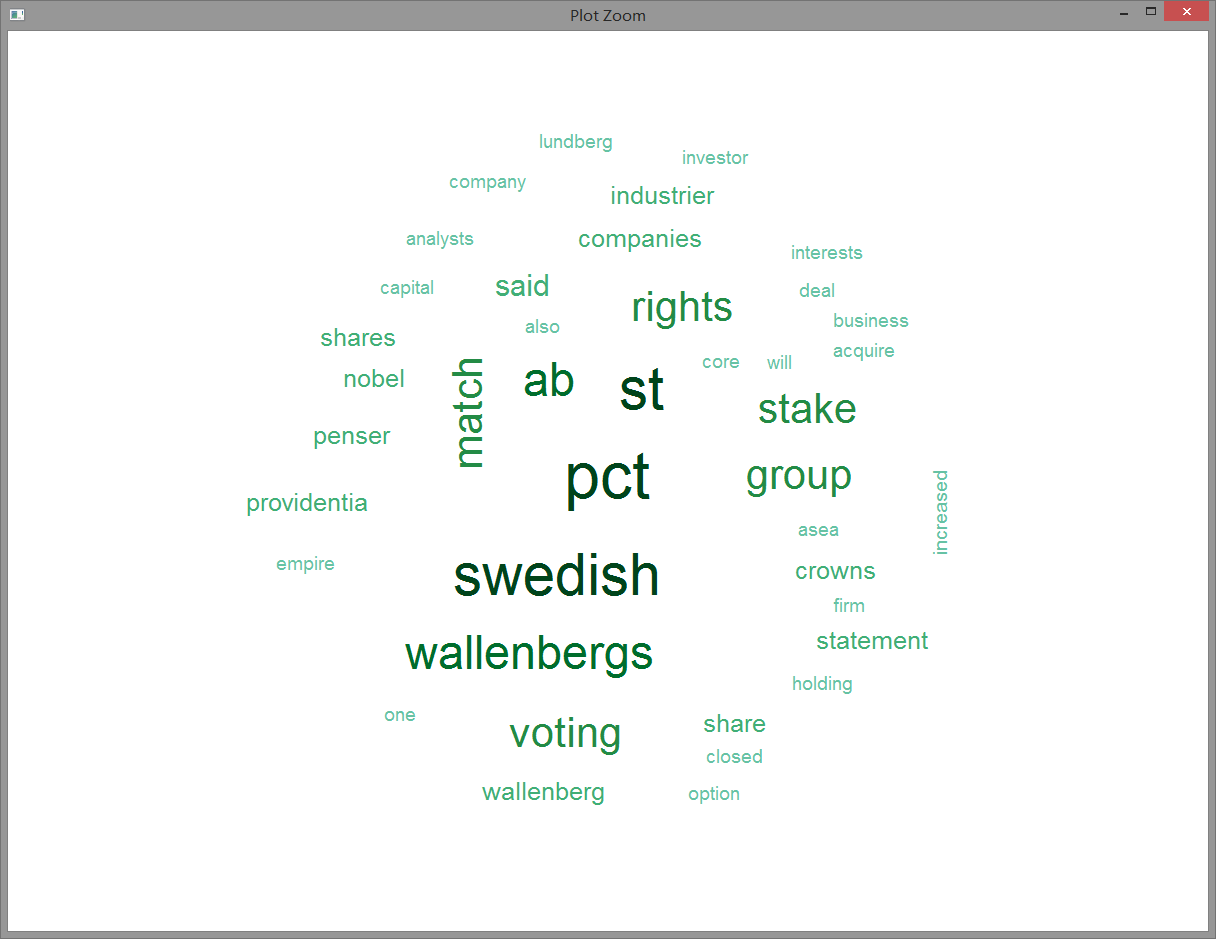
WordCloud:

**Discussion**: “dlrs” is the most frequently used term in this document.

**5. acq[19]**

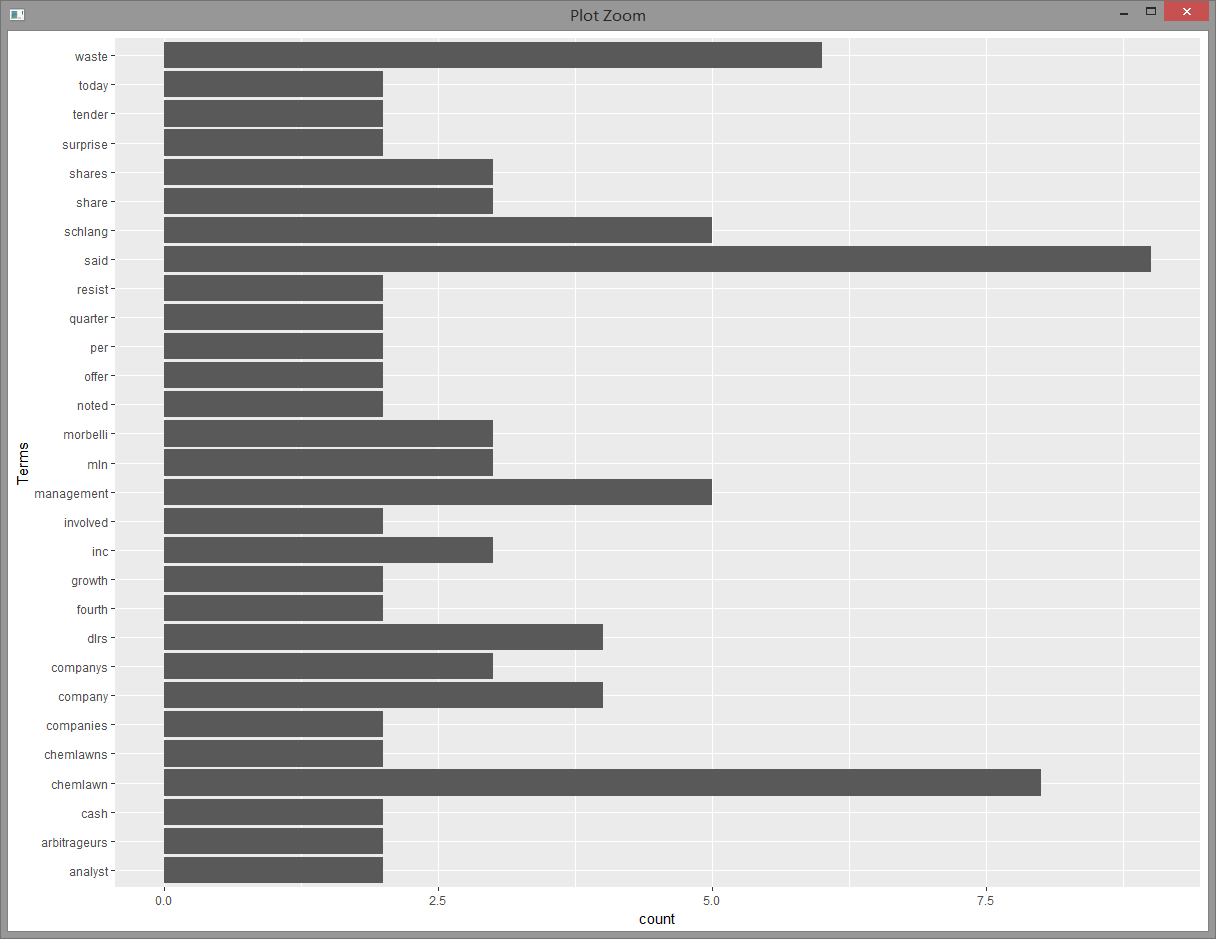
Word frequency:

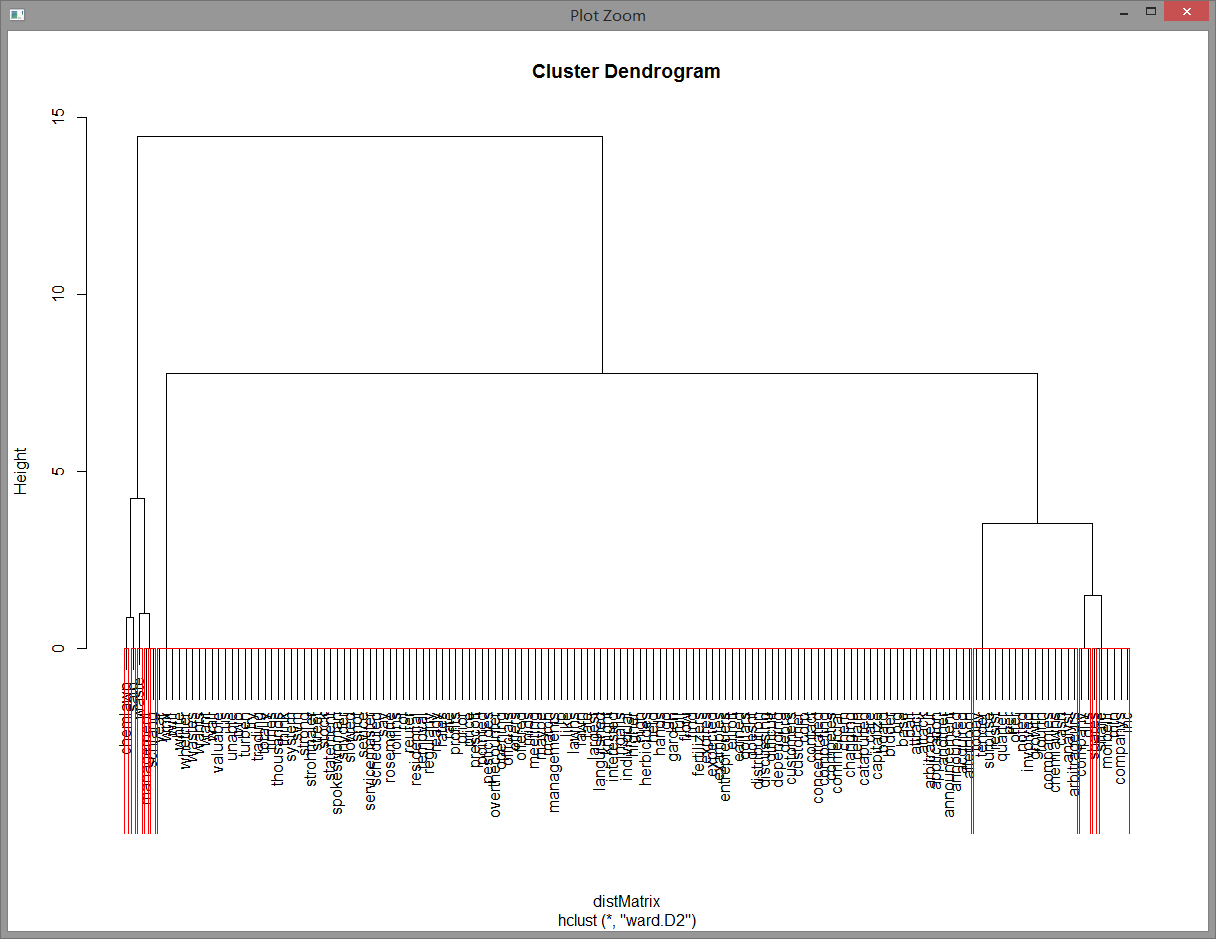
Dendrogram:

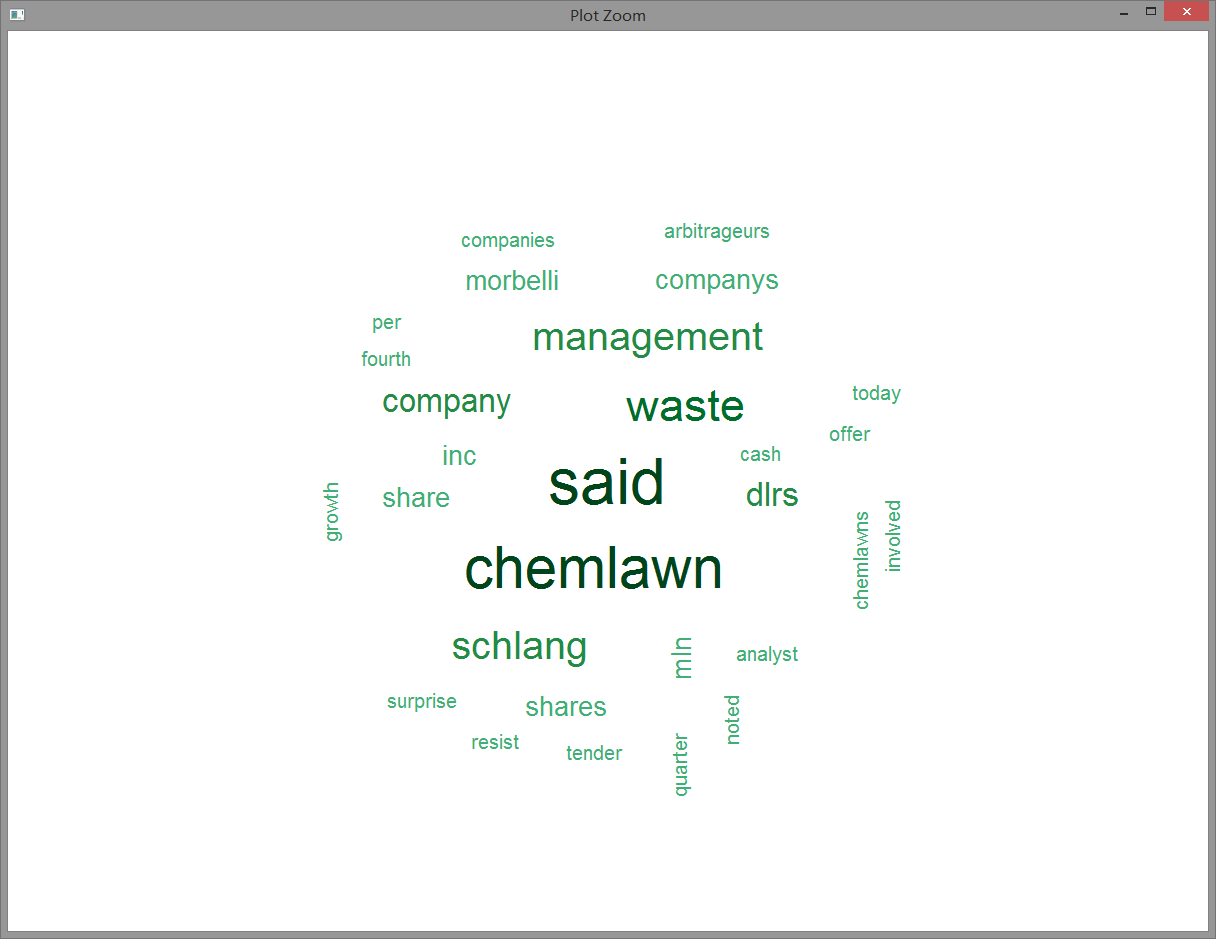
WordCloud:

**Discussion**: “st”, “pct” and “swedish” are the most frequently used terms in this document.

**6. acq[4]**

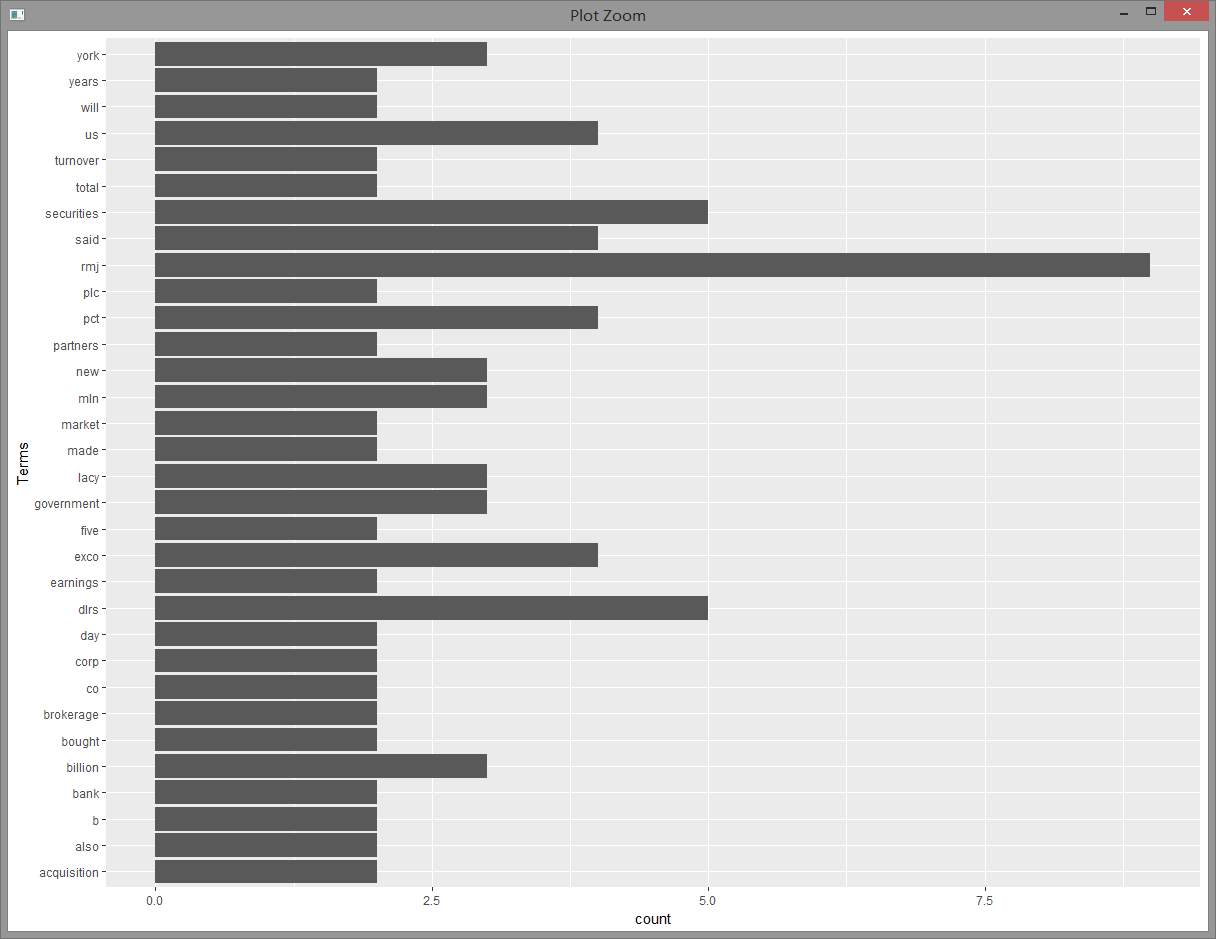
Word frequency:

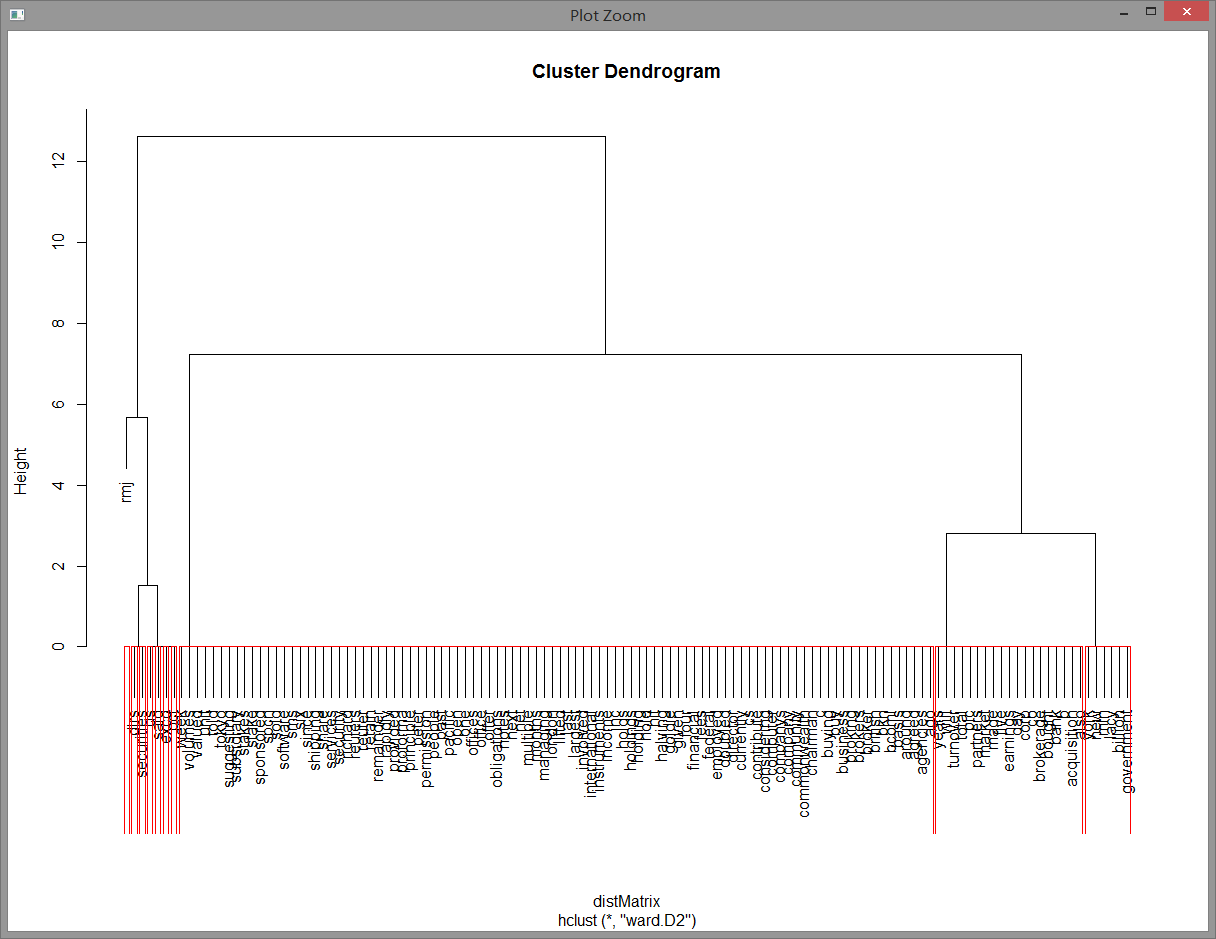
Dendrogram:

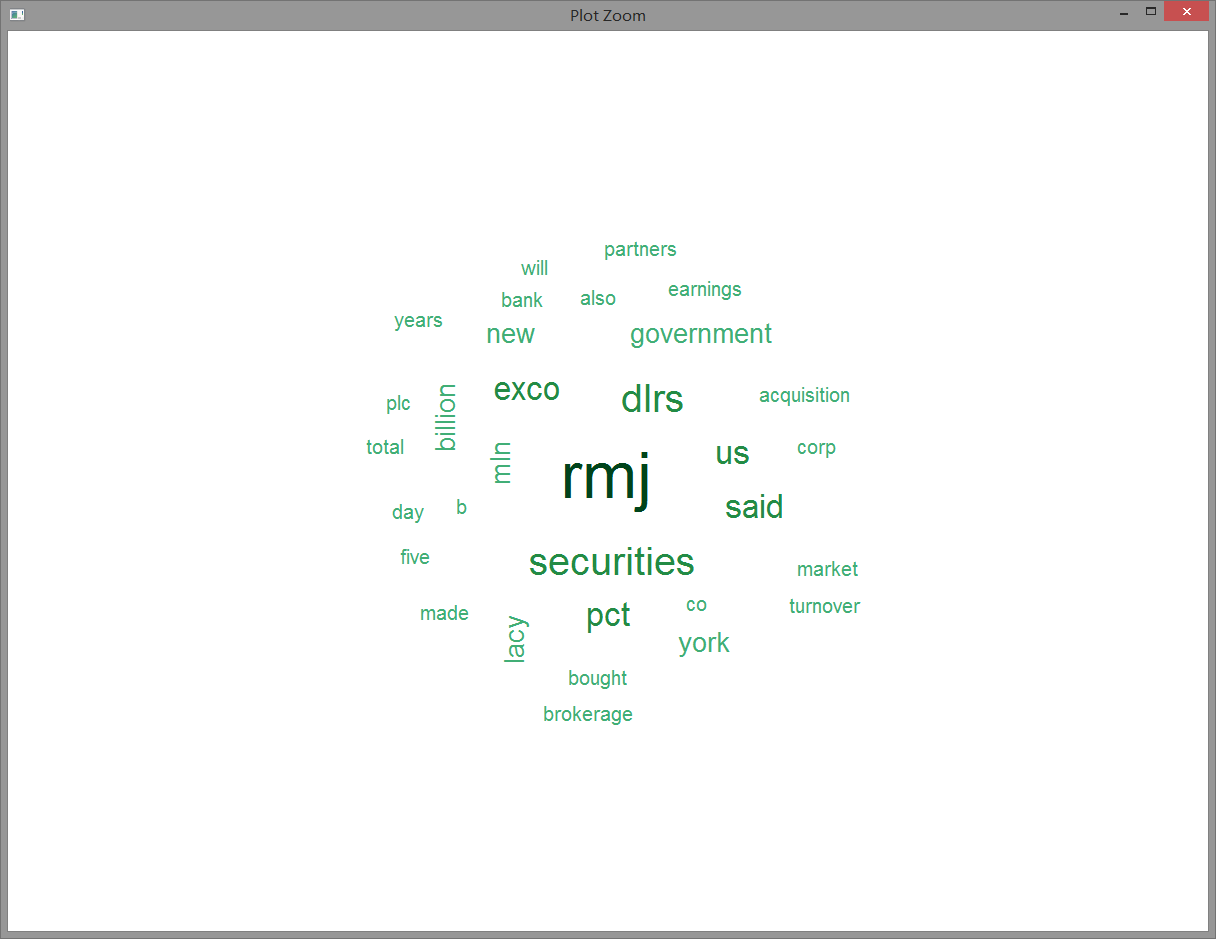
WordCloud:

**Discussion**: “said” and “chemlawn” are the most frequently used terms in this document.

**7. acq[22]**

Word frequency:

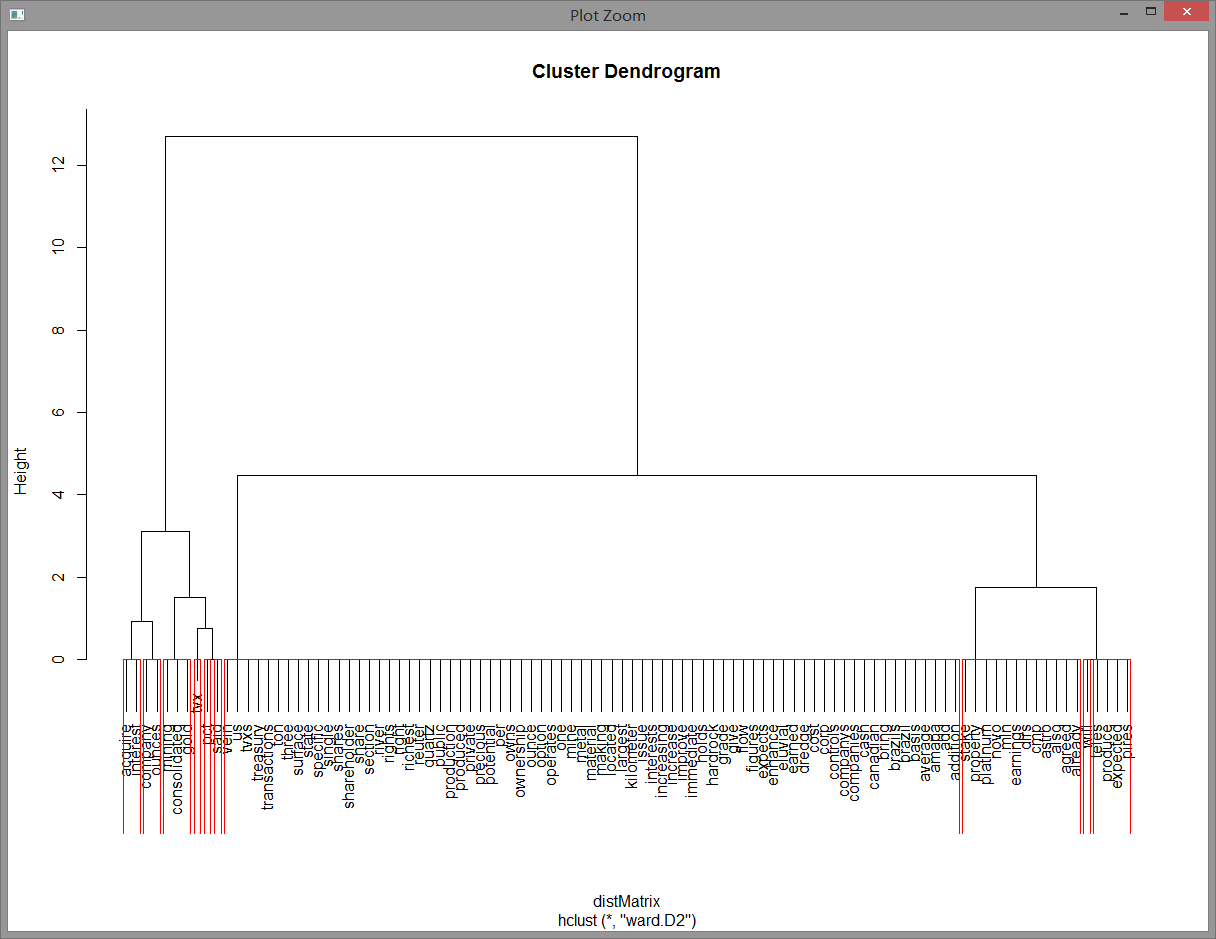
Dendrogram:

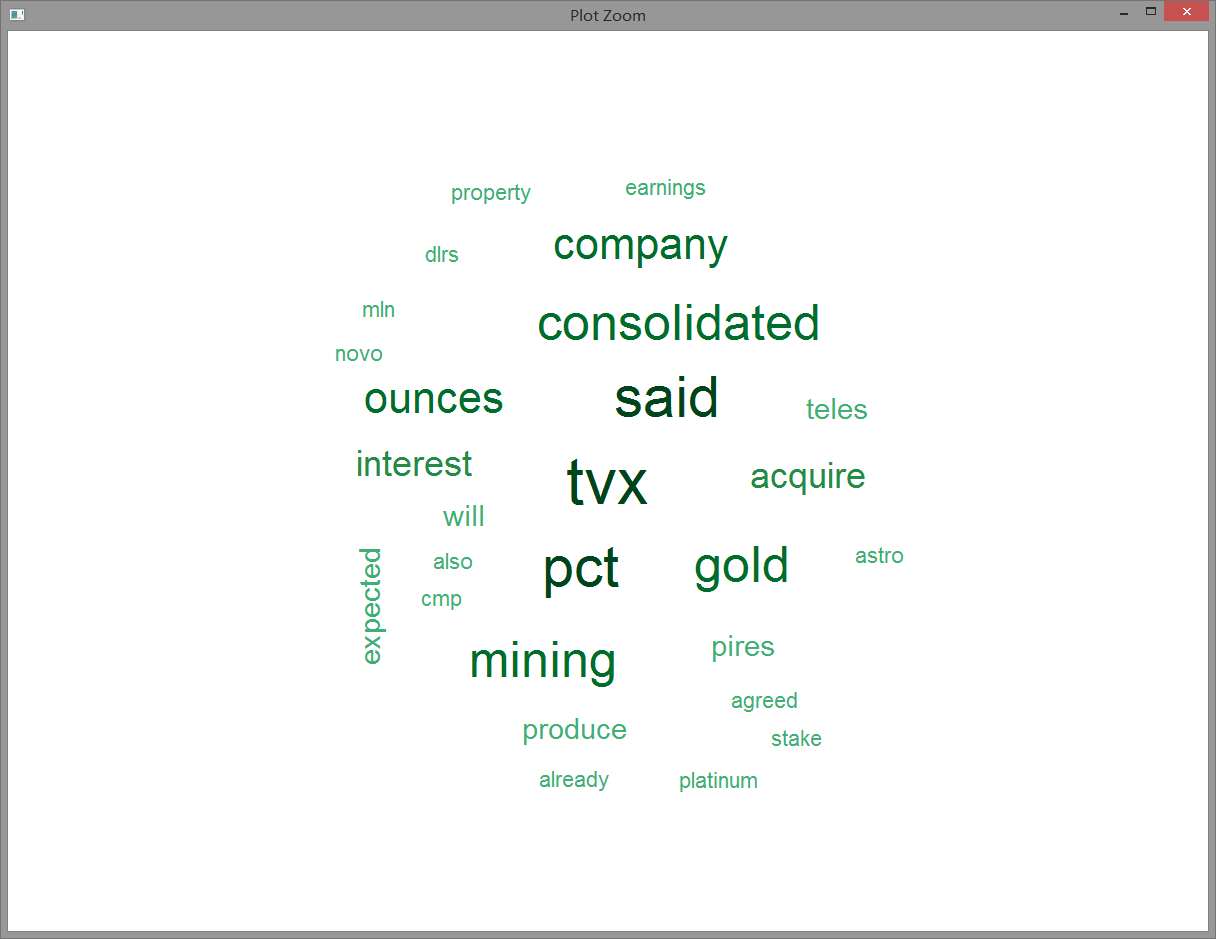
WordCloud:

**Discussion**: “rmj” is the most frequently used term in this document.

**8. acq[42]**

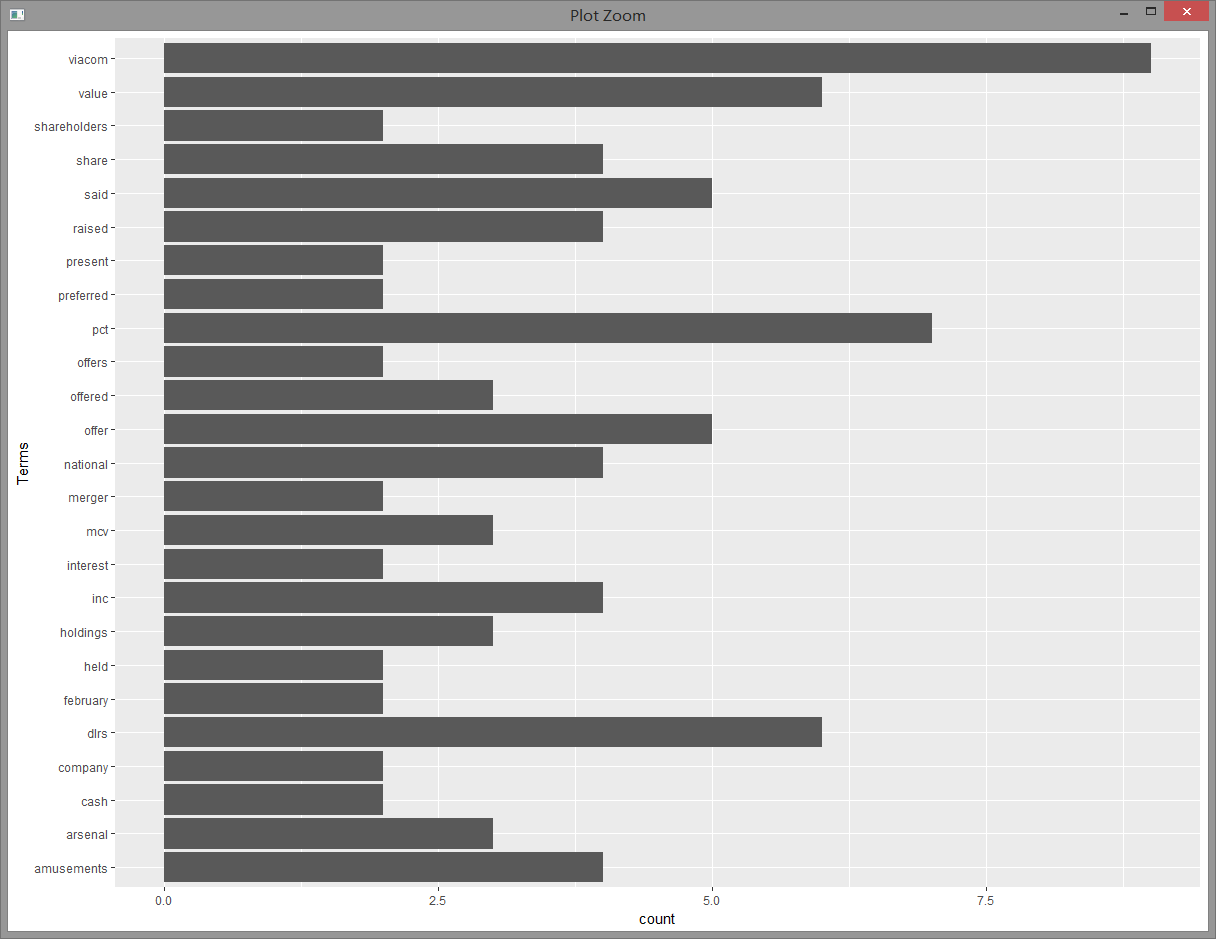
Word frequency:

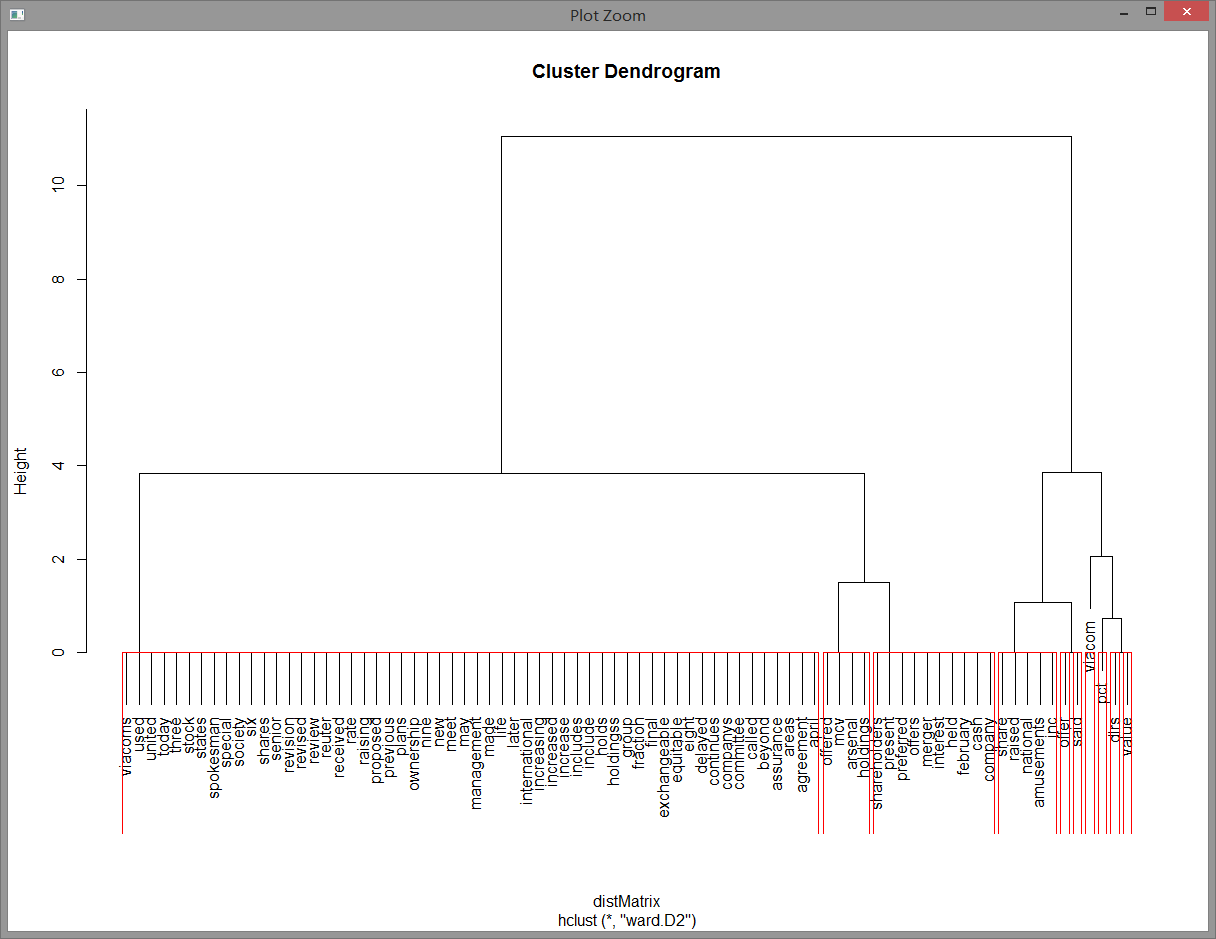
Dendrogram:

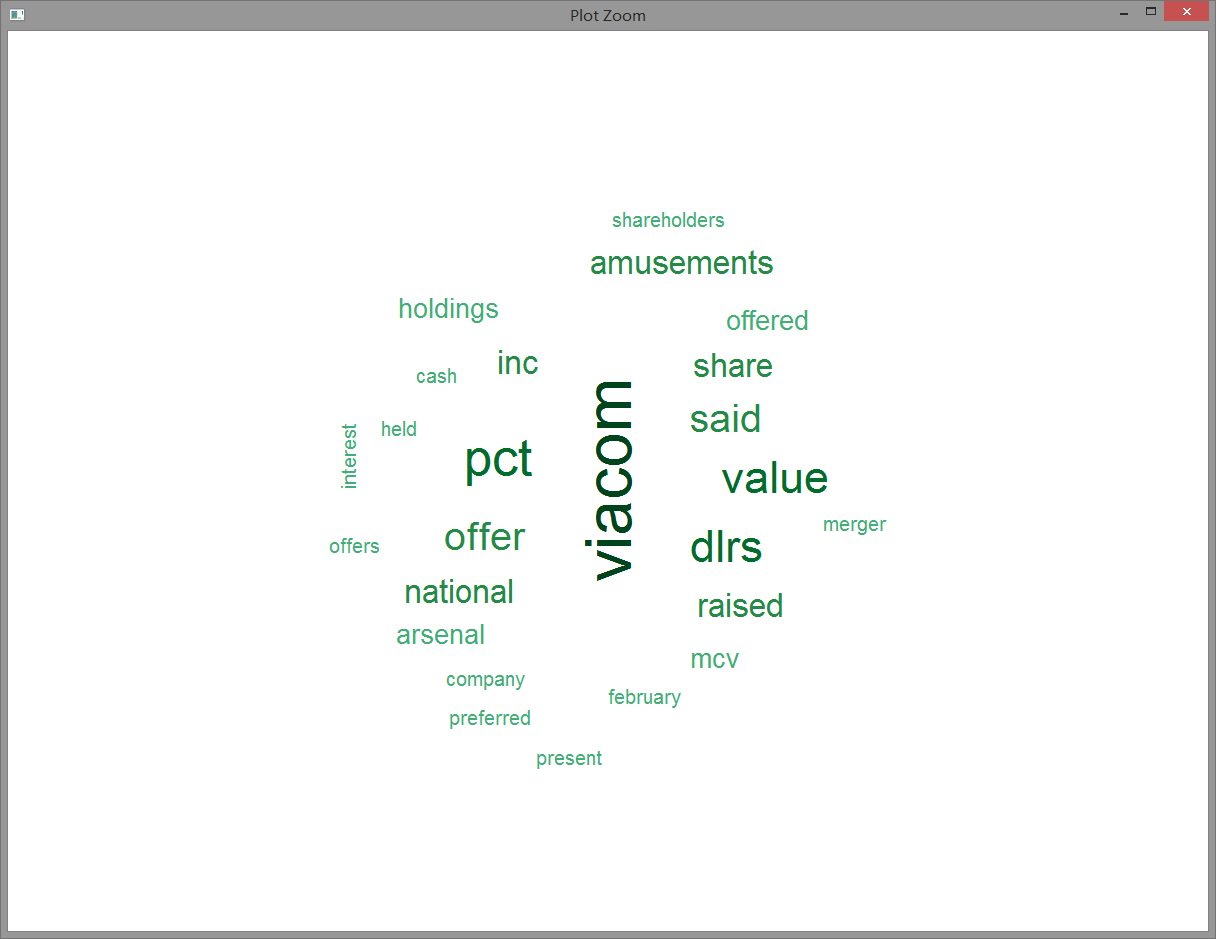
WordCloud:

**Discussion**: “said”, “tvx” and “pct” are the most frequently used terms in this document.

**9. acq[34]**

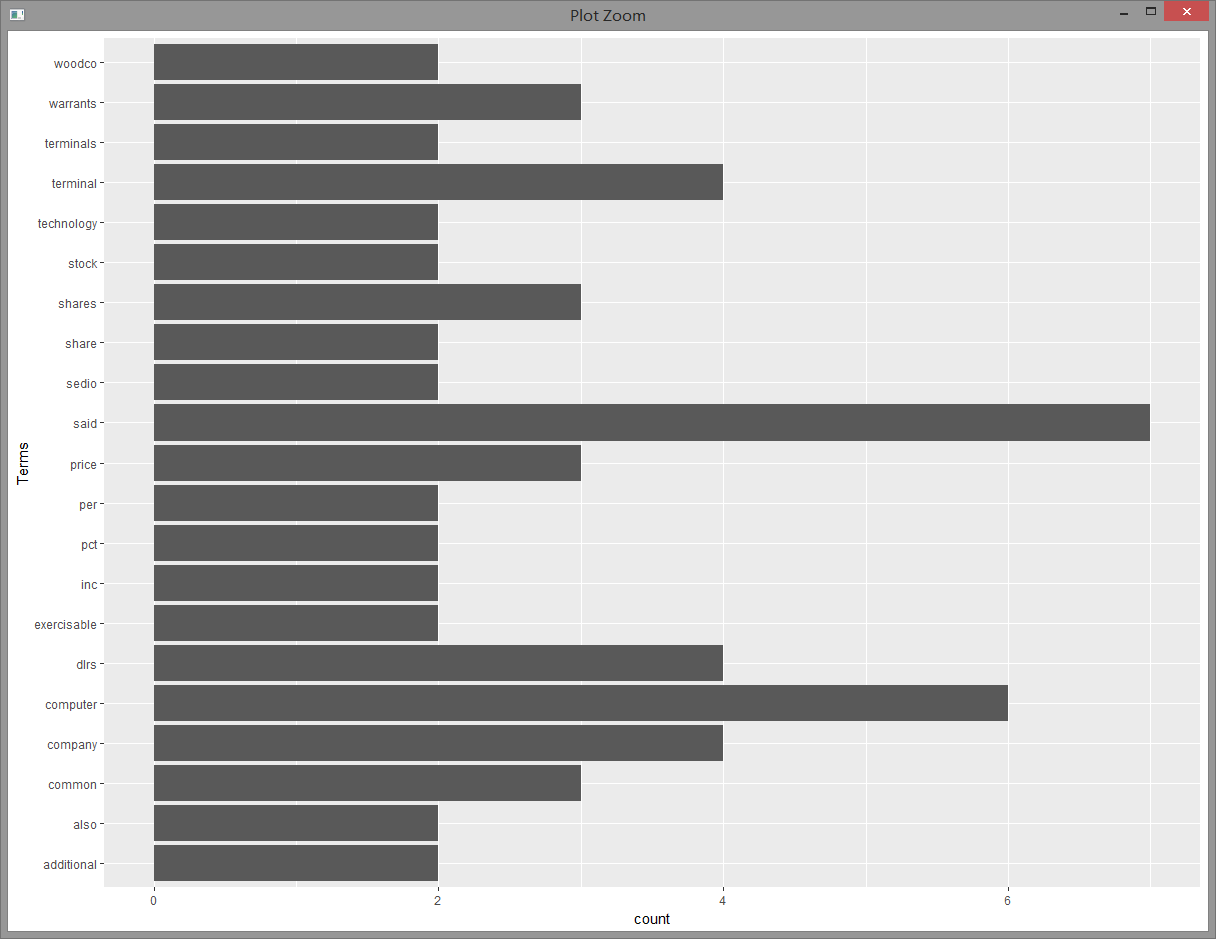
Word frequency:

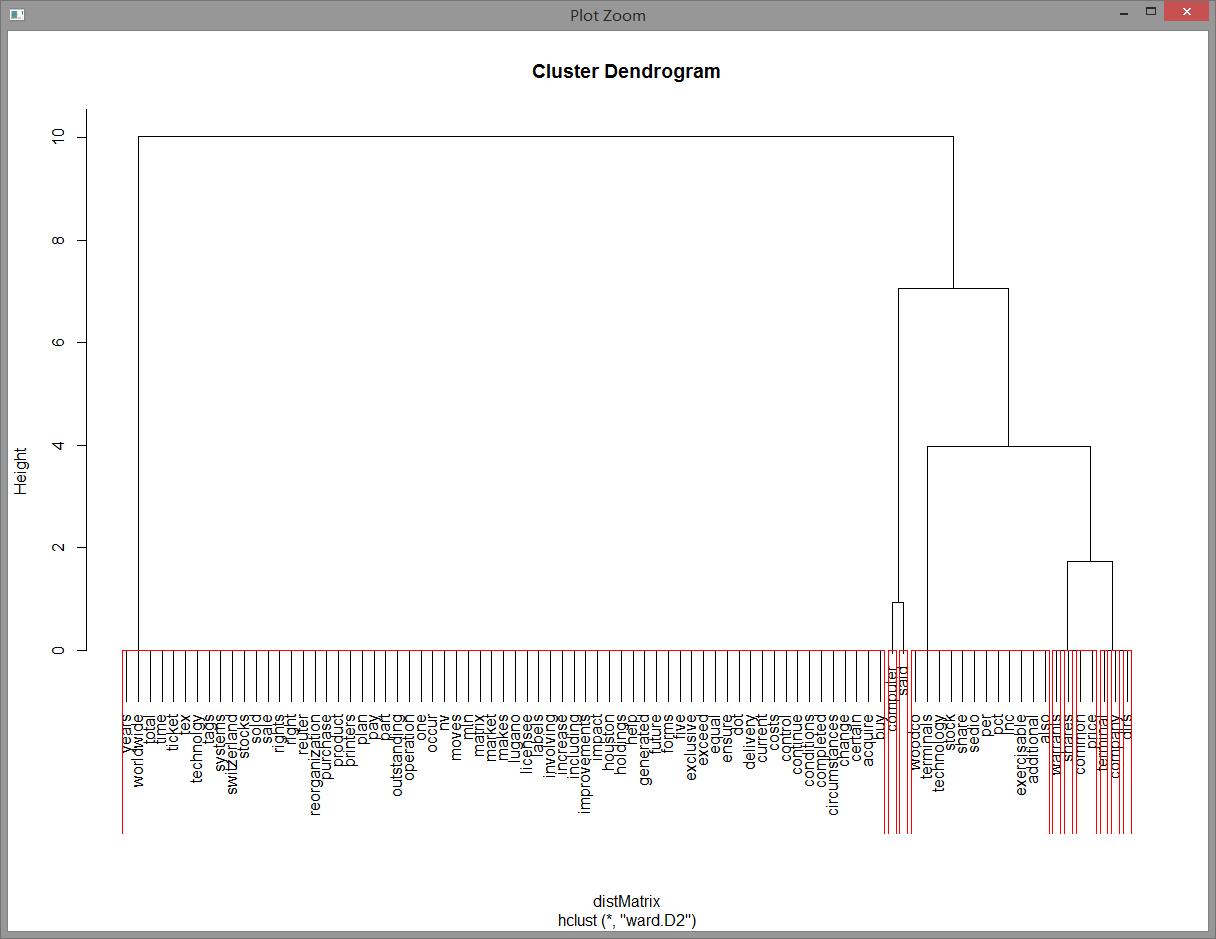
Dendrogram:

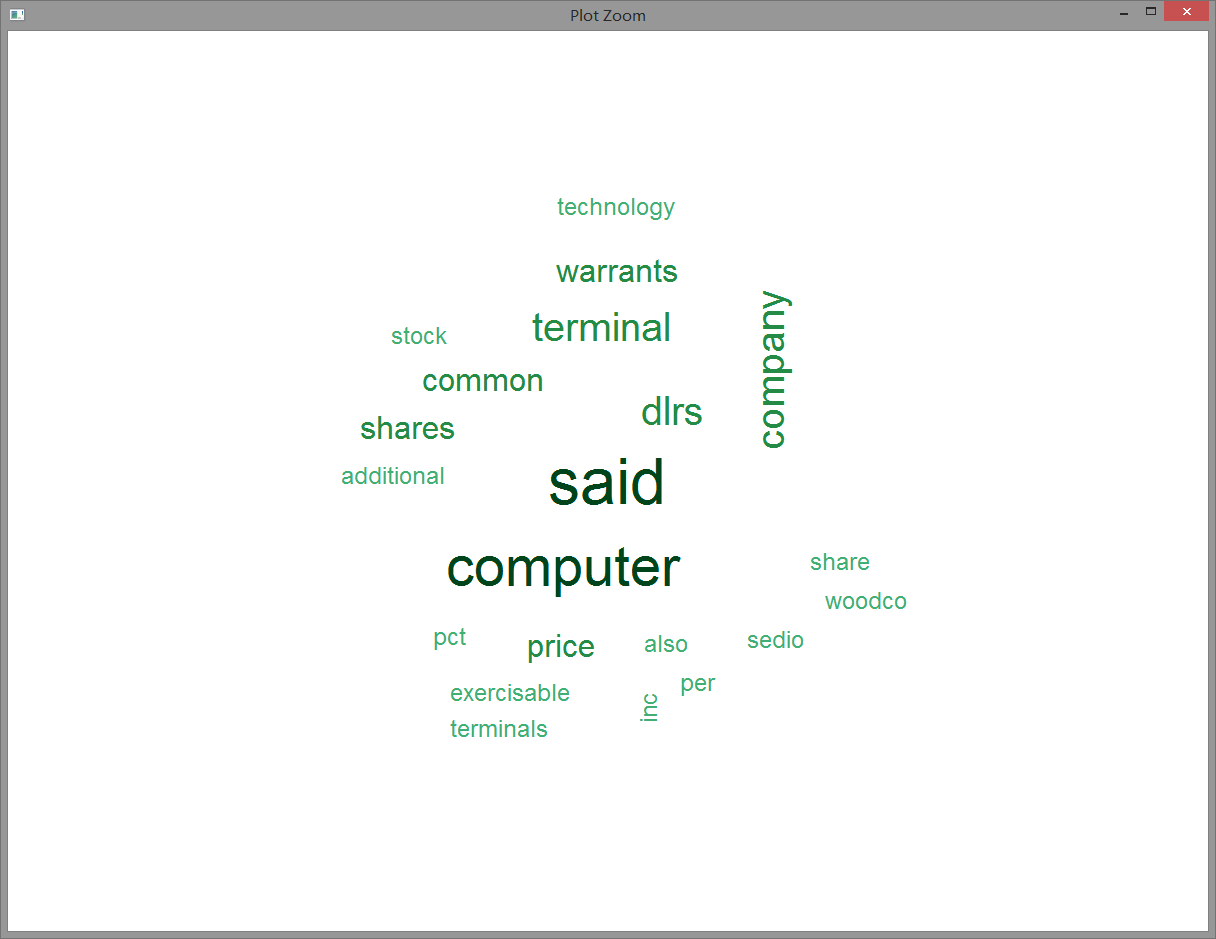
WordCloud:

**Discussion**: “viacom” is the most frequently used term in this document.

**10. acq[1]**

Word frequency:

Dendrogram:

WordCloud:

**Discussion**: “said” and “computer” are the most frequently used terms in this document.

**Longest word**: prudential-bache

**Longest sentence**: "american express co remained silent on market rumors it would spinoff all or part of its shearson lehman brothers inc but some analysts said the company may be considering such a move because it is unhappy with the market value of its stock"

d. Prior to removing the punctuation, find the longest word and longest sentence in each document from the 10 largest documents.

use acq[1] as the example.

find longest words:

# #store as words

# > acq1word=tokenize\_words(acq[[1]]$content);

# 

# #compute length of each words

# > acq1wordlen=array(dim = length(acq1word));

# > for(i in 1:length(acq1wordlen)){

# acq1wordlen[i]=nchar(acq1word[i]);

# }

# 

# #find the index of longest words

# > acq1wordindex=which(acq1wordlen==sort(acq1wordlen)[length(acq1wordlen)]);

# 

# #find the longest words

# > acq1Longestword=array(dim=length(acq1wordindex))

# > for(i in 1:length(acq1wordindex)){

# acq1Longestword[i]=acq1word[acq1wordindex[i]];

# }

# 

find longest sentence:

# #remove "\n"

# > removen <-function(x) gsub("\n", " ", x);

# > acq1sentence <- tm\_map(acq[1], content\_transformer(removen));

# 

# #store as sentences

# > acq1sen=tokenize\_sentences(acq1sentence[[1]]$content);

# 

# #compute length of each sentence

# > acq1senlen=array(dim = length(acq1sen))

# > for(i in 1:length(acq1senlen)){

# acq1senlen[i]=length(unlist(strsplit(acq1sen[i]," ")));

# }

# 

# #find the index of longest sentences

# > acq1senindex=which(acq1senlen==sort(acq1senlen)[length(acq1senlen)]);

# 

# #find the longest sentences

# > acq1Longestsen=array(dim = length(acq1senindex))

# > for(i in 1:length(acq1senindex)){

# acq1Longestsen[i]=acq1sen[acq1senindex[i]];

# }

# 

**acq[7]:**

longest words: “capitalization".

longest sentences: "american express co remained silent on market rumors it would spinoff all or part of its shearson lehman brothers inc but some analysts said the company may be considering such a move because it is unhappy with the market value of its stock.”

**acq[25]:**

longest words: "reorganization" and "unsuccessfully"

longest sentences: "american express co rumored to be considering a spinoff of part of shearson lehman brothers inc said it is studying a range of options for its brokerage unit that could improve shearon s access to capital and help it meet broadening international competition.”

**acq[29]:**

longest words: "shareholders" and "commencement"

longest sentences: "if all the shares of purolator are tendered shareholders would receive for each share 29 dlrs cash six dlrs in debentures and a warrant to buy shares in a subsidiary of pc acquisition containing the u s courier operations.”

**acq[47]:**

longest words: “confidentiality"

longest sentences: "the redstone group which has a 19 5 pct stake in viacom and the management group which has a 5 4 pct stake have both agreed not to buy more shares of the company until a merger is completed unless the purchases are part of a tender offer for at least half of the outstanding stock.”

**acq[19]:**

longest words: "concentrating"

longest sentences: "but analysts say the wallenbergs position in the electrical engineering firm asea ab <asea st> is also too small at 12 6 pct of the voting rights and there has been growing speculation that the group will be forced to sell off fringe interests to protect its core activities"

**acq[4]:**

longest words: "servicemaster"

longest sentences: "shares of chemlawn shot up 11 5 8 to 29 3 8 in over the counter trading with 3 8 mln of the company s 10 1 mln shares changing hands by late afternoon.”

**acq[22]:**

longest words: "international"

longest sentences:

1. “<exco international plc> a subsidiary of british and commonwealth shipping co plc <bcom l> said it had agreed in principle to buy an 80 pct stake in <rmj holdings corp> for about 79 mln dlrs."

2. "exco chairman richard lacy told reuters the acquisition was being made from bank of new york co inc <bk n> which currently holds a 50 1 pct and from rmj partners who hold the remainder.”

**acq[42]:**

longest words: "consolidated" and "transactions"

longest sentences: "<consolidated tvx mining corp> said it agreed to issue 7 8 mln treasury shares to acquire interests in three gold mining companies in brazil and an option to increase the company s interest in a platinum property.”

**acq[34]:**

longest words: "international"

longest sentences: "viacom said mcv holdings a group which includes the company s senior management and the equitable life assurance society of the united states raised the value of its offer by increasing the value of the preferred being offered to 8 50 dlrs from 8 00 dlrs a share and raising the ownership in the new company to be held by present viacom shareholders to 45 pct from 25 pct.”

**acq[1]:**

longest words: "reorganization"

longest sentences:

[1] "computer terminal systems inc said it has completed the sale of 200 000 shares of its common stock and warrants to acquire an additional one mln shares to <sedio n v > of lugano switzerland for 50 000 dlrs"

[2] "computer terminal said sedio also has the right to buy additional shares and increase its total holdings up to 40 pct of the computer terminal s outstanding common stock under certain circumstances involving change of control at the company"

**e. Print a table of the length of each sentence in each of the 10 documents.**

note the number of sentences in these 10 docs is at most 31.

# #index of 10 longest docs

# > countindex

# 

# #compute number of words of each sentence

# > NumOfWords<-matrix(nrow=10,ncol=31);

# > for(i in 1:10){

# #remove "\n"

# removen <-function(x) gsub("\n", " ", x);

# sentence <- tm\_map(acq[countindex[i]], content\_transformer(removen));

# #store as sentences

# sen=tokenize\_sentences(sentence[[1]]$content);

# #compute number of words for sentence j

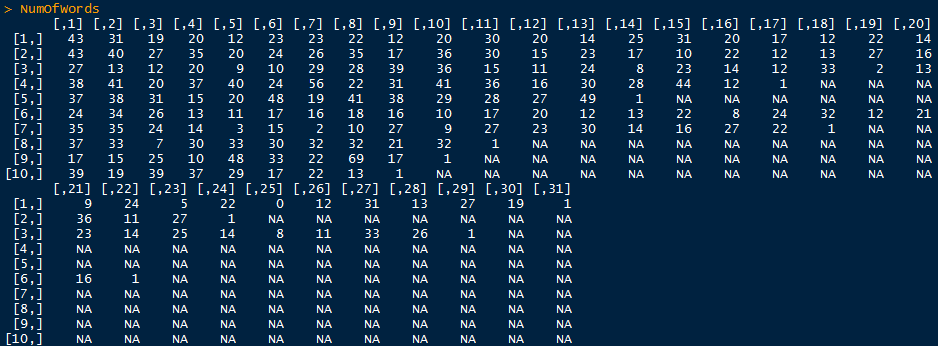
# for(j in 1:length(sen)){

# NumOfWords[i,j]=length(unlist(strsplit(sen[j]," ")));

# }

# }

table of the length of each sentence in each of the 10 documents:



f. For each sentence of each document, remove the punctuation. Display the sentences.

Also use acq[1] as the example

#remove punctuation

> removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]\*", "", x)

> acq.rem <- tm\_map(acq[1], content\_transformer(removeNumPunct))

#remove "\n"

> removen <-function(x) gsub("\n", " ", x);

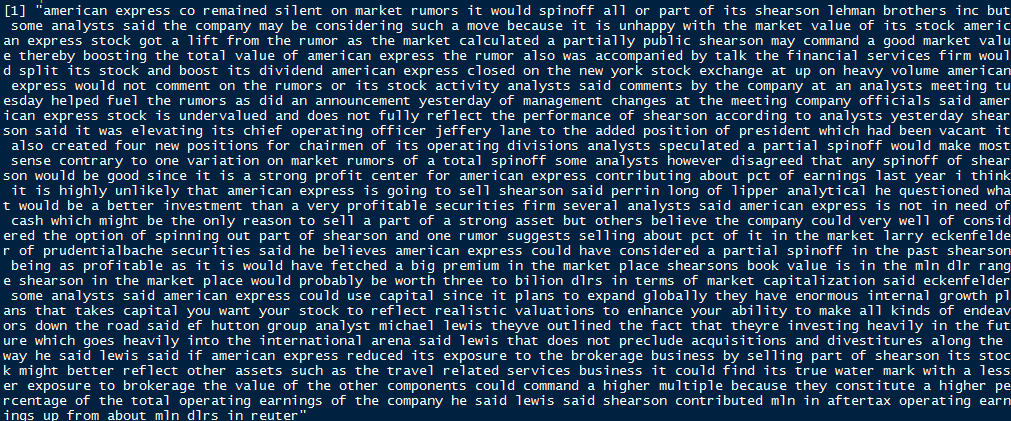
> acq.rem2 <- tm\_map(acq.rem, content\_transformer(removen));

# > acq.rem2=tokenize\_sentences(acq.rem2[[1]]$content);

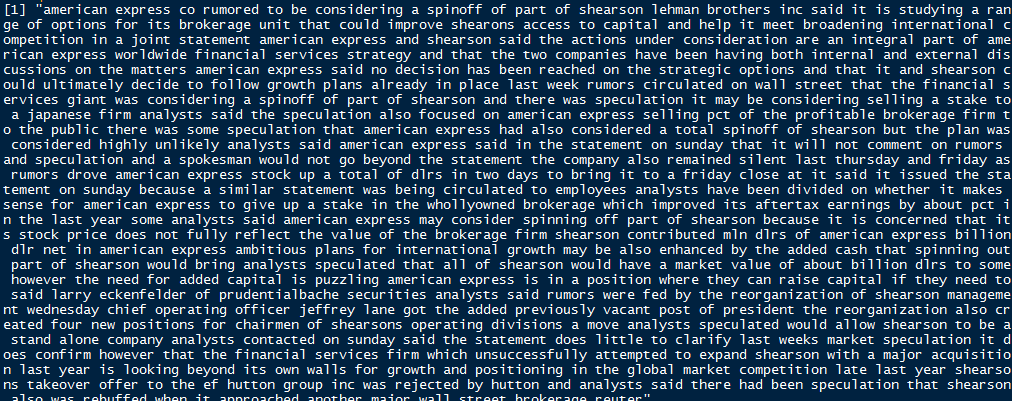
# 

Display sentences:

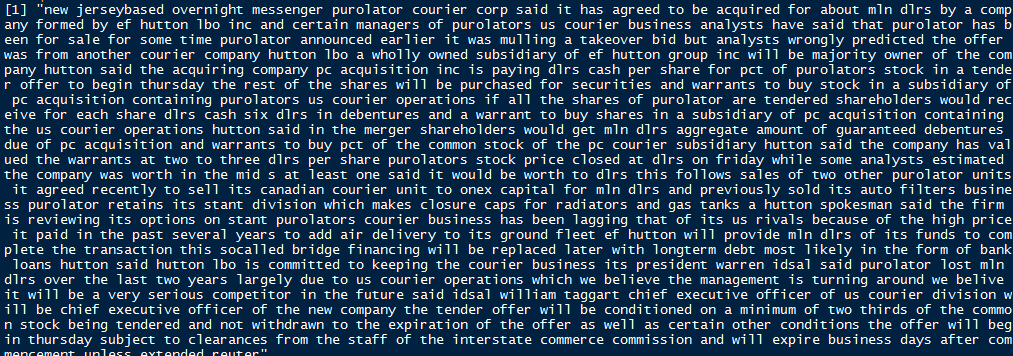
**acq[7]:**



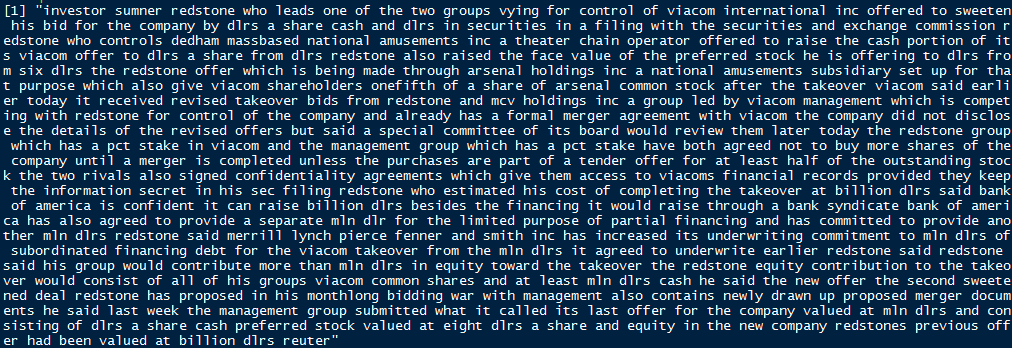
**acq[25]:**



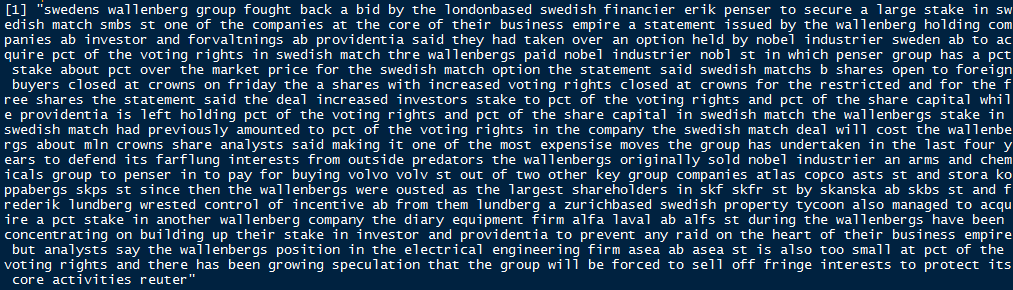
**acq[29]:**



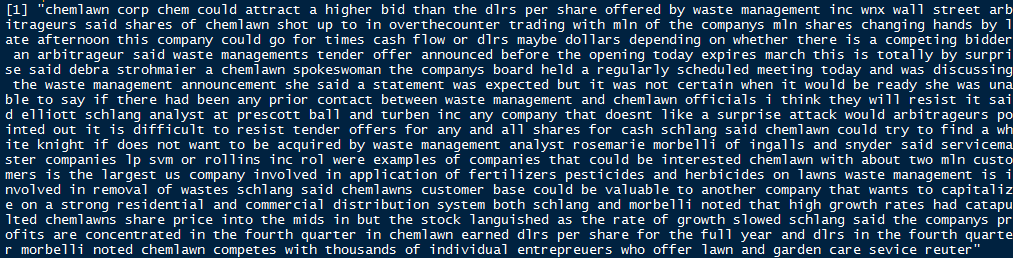
**acq[47]:**



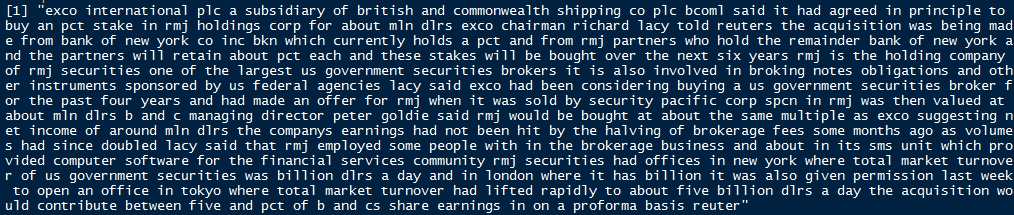
**acq[19]:**



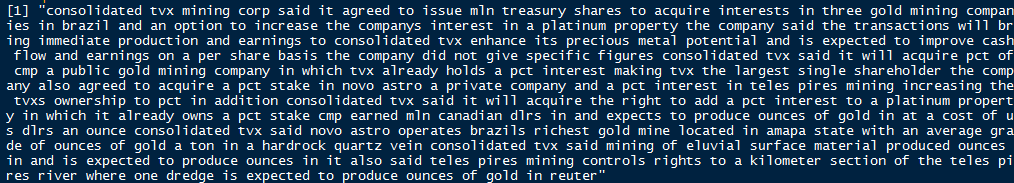
**acq[4]:**



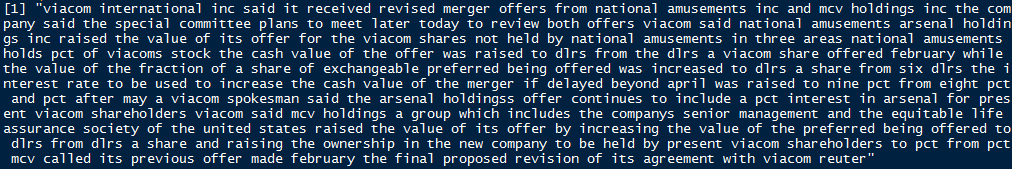
**acq[22]:**



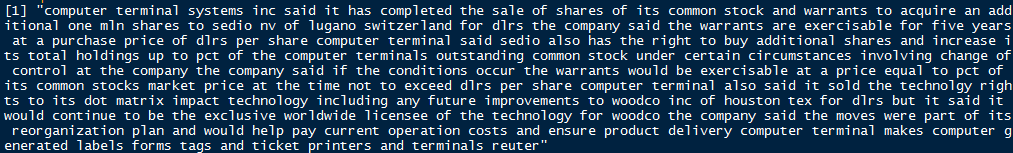
**acq[42]:**



**acq[34]:**



**acq[1]:**



g. For each word print its part of speech using the Wordnet package.

First, we downloaded the wordnet database wn3.1.dict from the website: <http://wordnet.princeton.edu/wordnet/download/current-version/>

use acq[1] as example

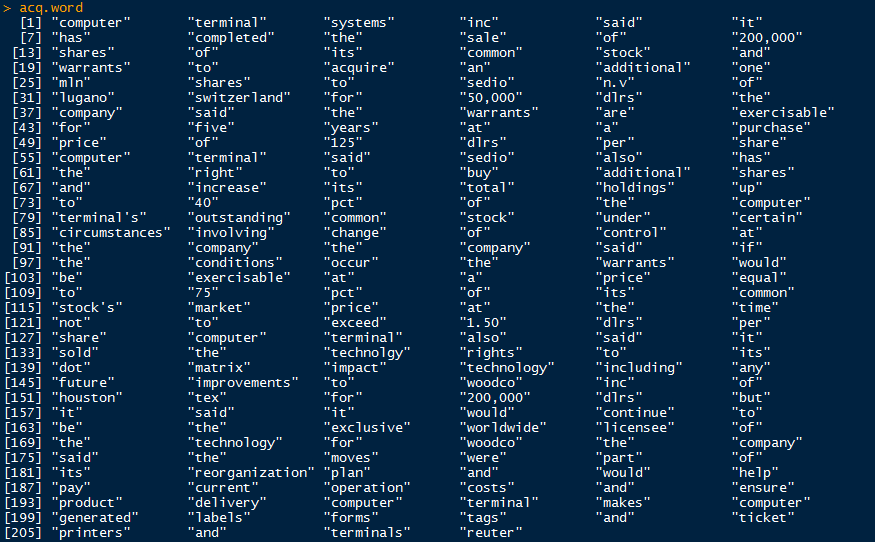
#init the dict, set the path

> initDict("D:/R/R-3.3.1/R-3.3.1/library/wn3.1.dict/dict")

> setDict("D:/R/R-3.3.1/R-3.3.1/library/wn3.1.dict/dict")

#store as words

# > acq.word=tokenize\_words(acq[[1]]$content);



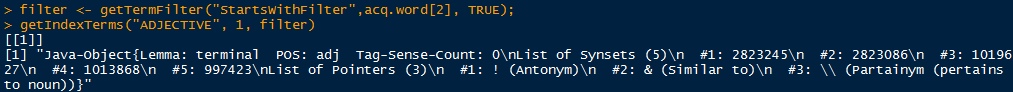
#set the filter and get the result

# > filter <- getTermFilter("StartsWithFilter",acq.word[1], TRUE);

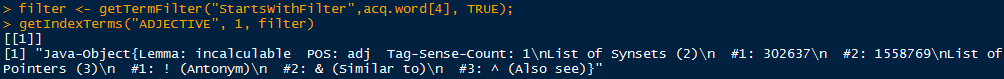
# > getIndexTerms("ADJECTIVE", 1, filter)

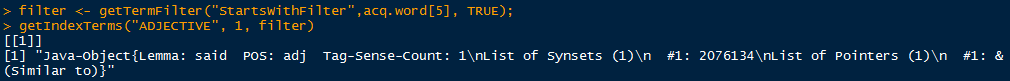
# 

more examples:









**getTermFilter()：**Get a term filter.

syntax: getTermFilter(type, word, ignoreCase)

parameters：

type: Filter type. Available filters ( “ContainsFilter", "EndsWithFilter", "ExactMatchFilter", “RegexFilter", "SoundFilter", "StartsWithFilter", and “WildcardFilter".) Can also be a unique abbreviation of an available filter name.

word: Term to be matched.

ignoreCase: Indicates whether lower and upper case are distinguished.

**getIndexTerms()**：Get index terms from a WordNet dictionary as specified by a filter.

syntax：getIndexTerms(pos, *m*axLimit, filter)

parameters：

pos: Part of speech type. Must be either "ADJECTIVE", "ADVERB", "NOUN", or "VERB".

maxLimit: Maximum number of results.

filter: A term filter

h. Analyze word frequency using functions from package zipfR.

**spc()**：spc objects are used to represent a word frequency spectrum (either an observed spectrum or the expected spectrum of a LNRE model at a given sample size).

use acq[1] as example:

# > SATlow <- tm\_map(acq[1], content\_transformer(tolower))

# > removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]\*", "", x)

# > SATcl <- tm\_map(SATlow, content\_transformer(removeNumPunct))

# > myStopwords <- c(stopwords('english'))

# > SATstop <- tm\_map(SATcl, removeWords, myStopwords)

# > SATtdm2 <-TermDocumentMatrix(SATstop,control=list(wordLengths=c(1,Inf)));

# > term.freq<-rowSums(as.matrix(SATtdm2));

# > term.freq<-subset(term.freq,term.freq>=2);

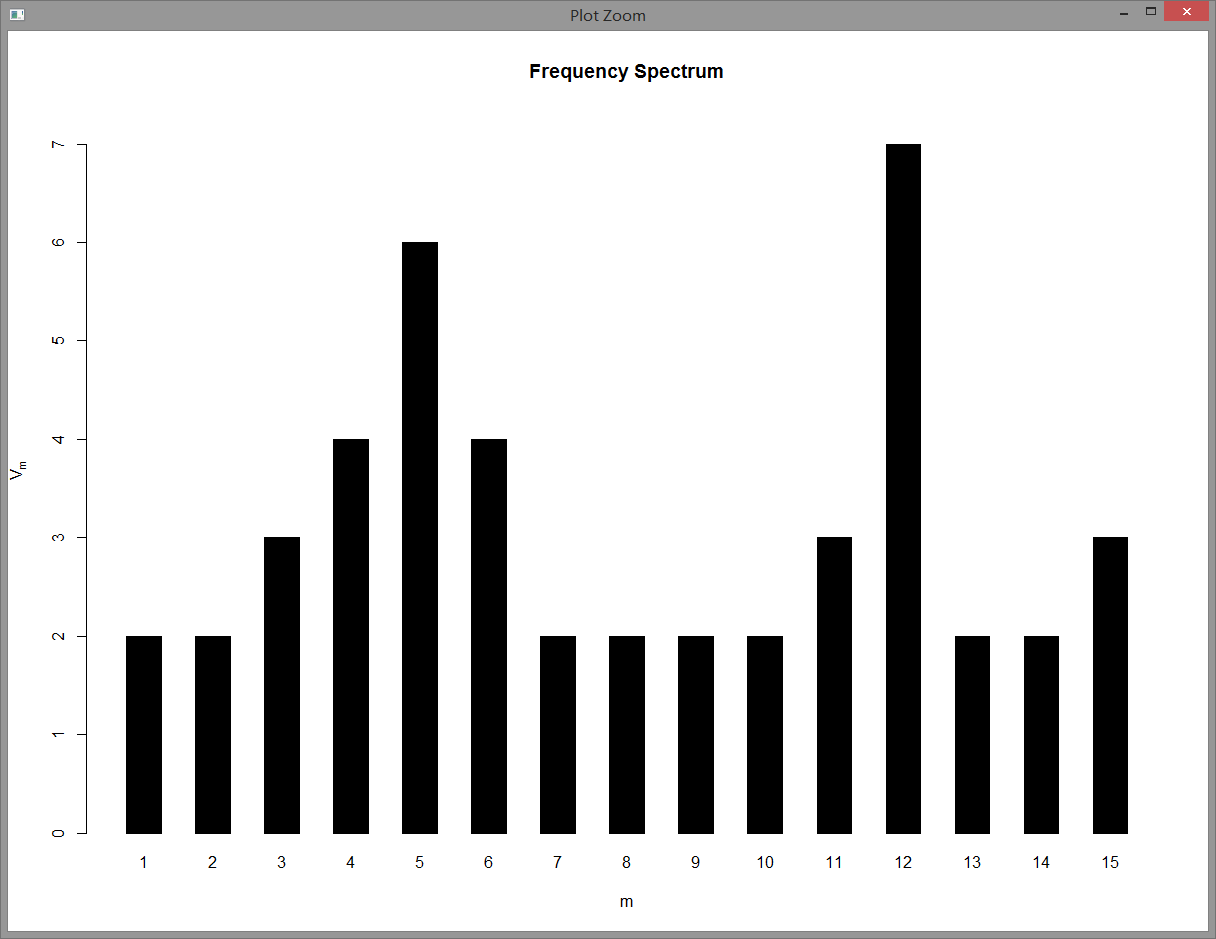
> term.freq.df=as.data.frame(term.freq)

> term.freq.dfrn=row.names(term.freq.df)

> term.freq.wordno=1:length(term.freq.dfrn)

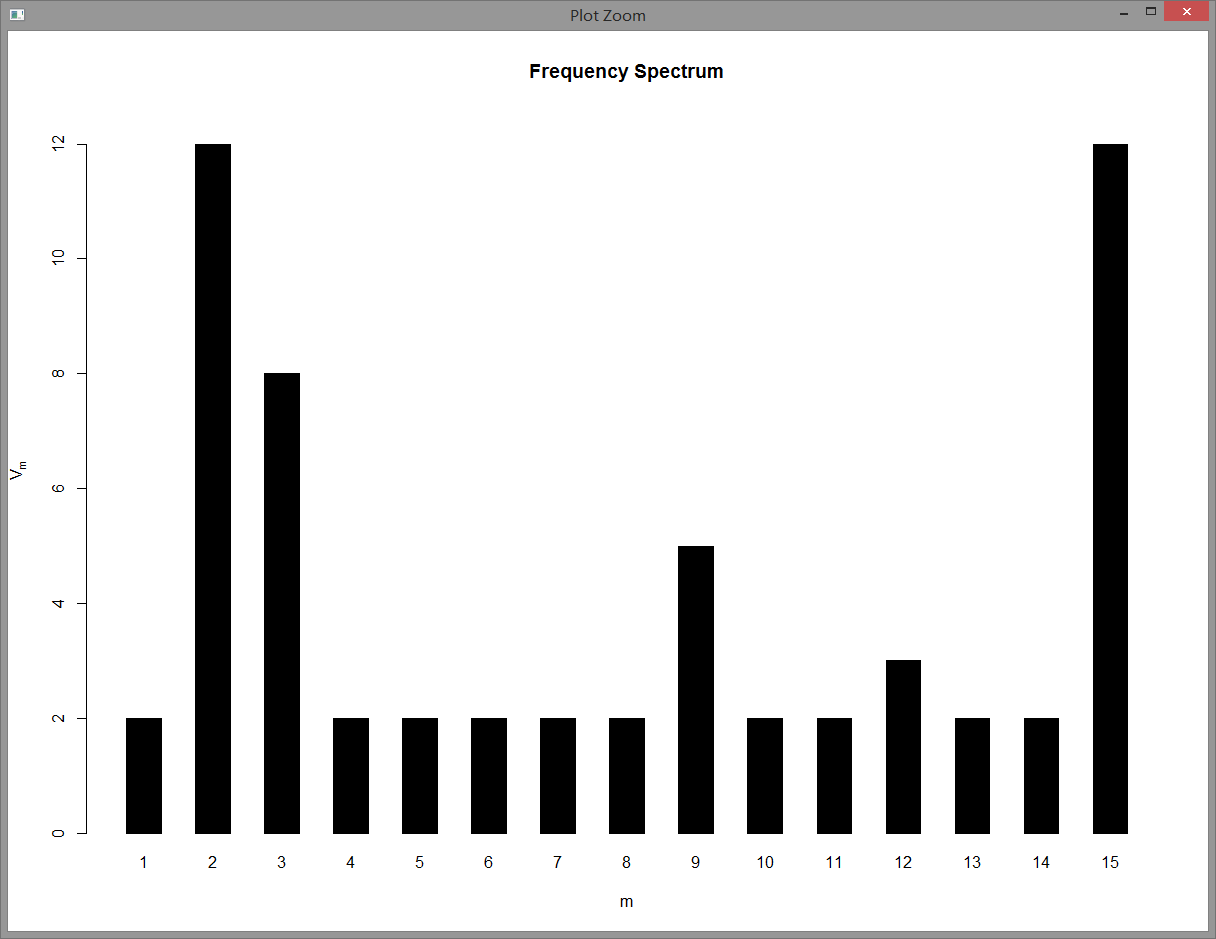
> term.freq.spc=spc(term.freq,term.freq.wordno)

> plot(term.freq.spc)

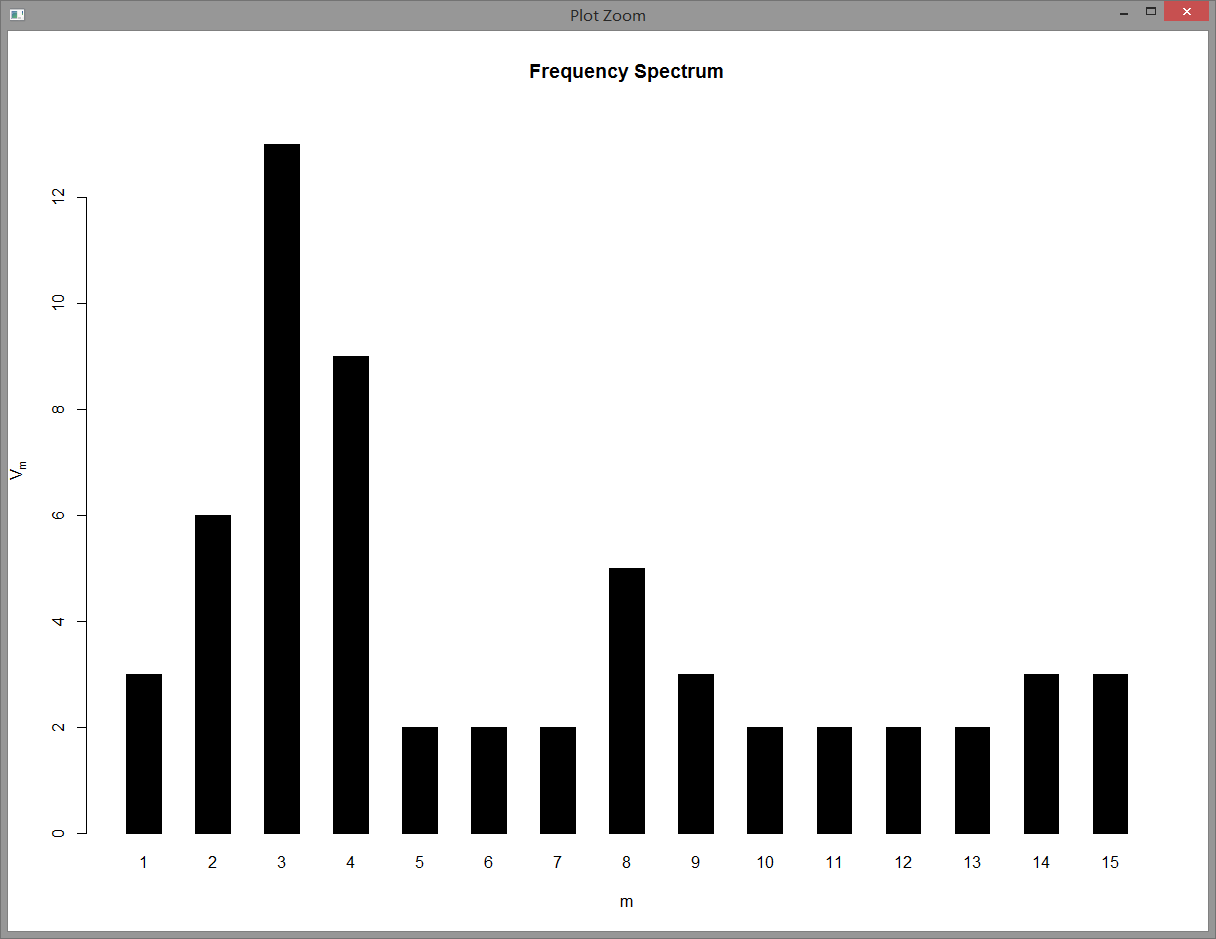


The frequency Spectrum for the 10 longest documents:

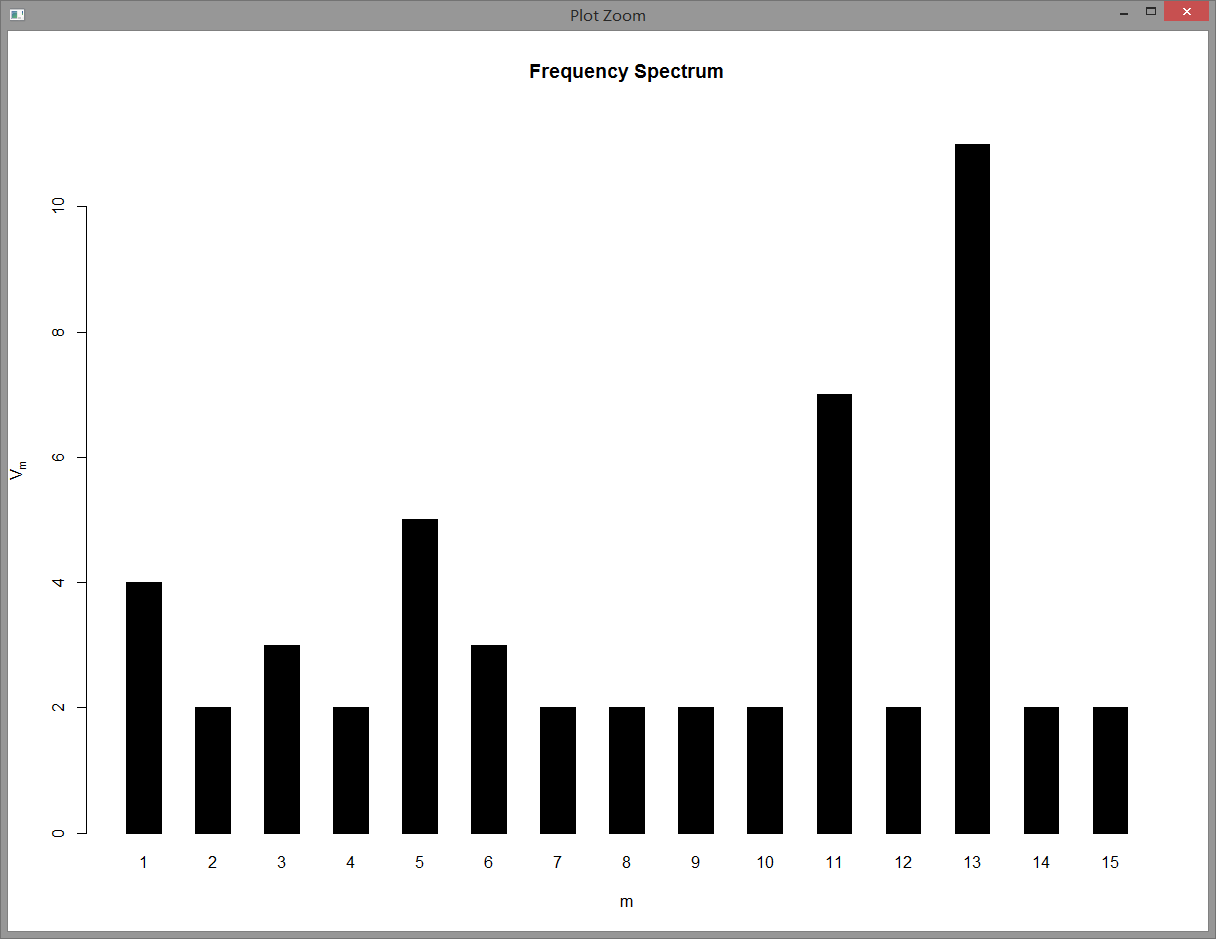
**acq[7]**



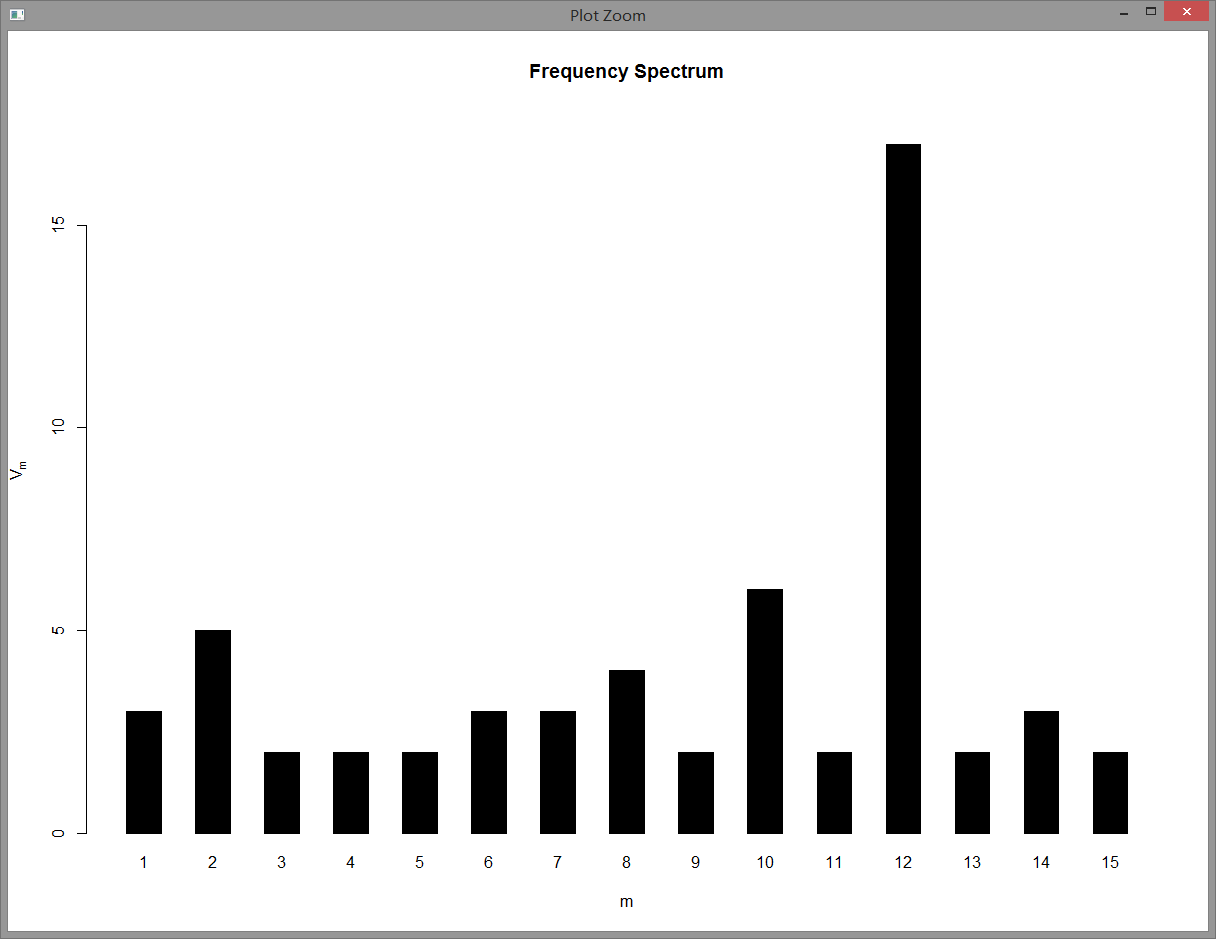
**acq[25]**



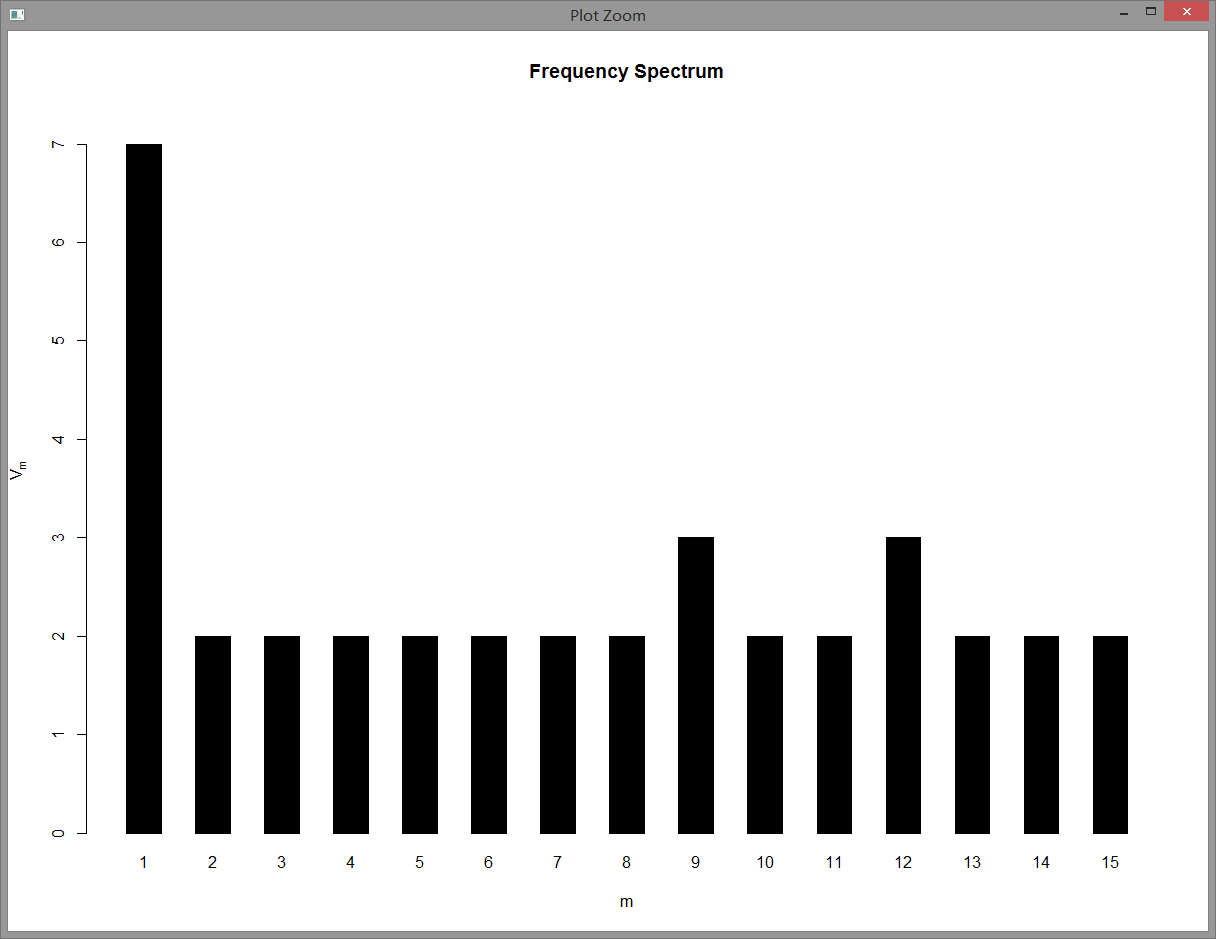
**acq[29]**



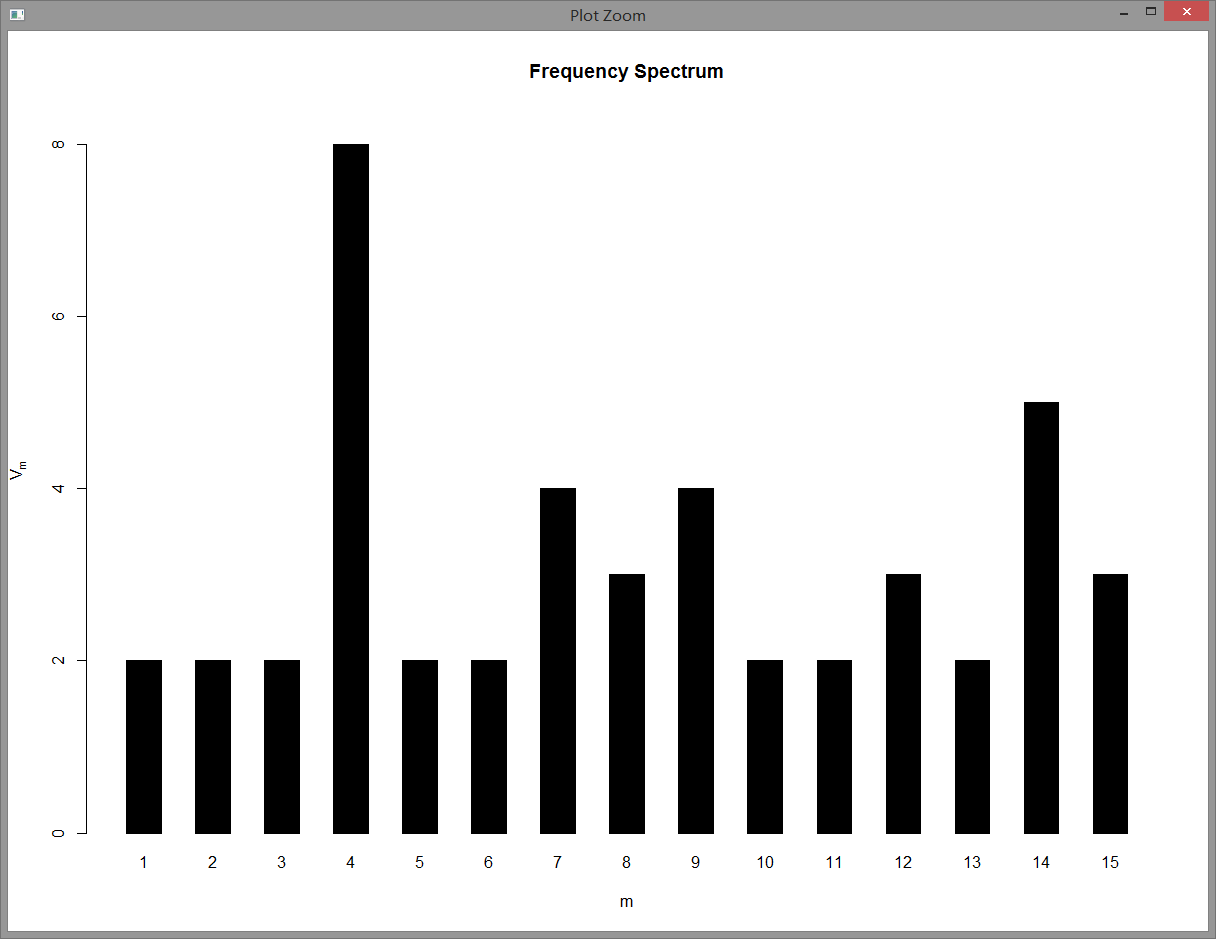
**acq[47]**



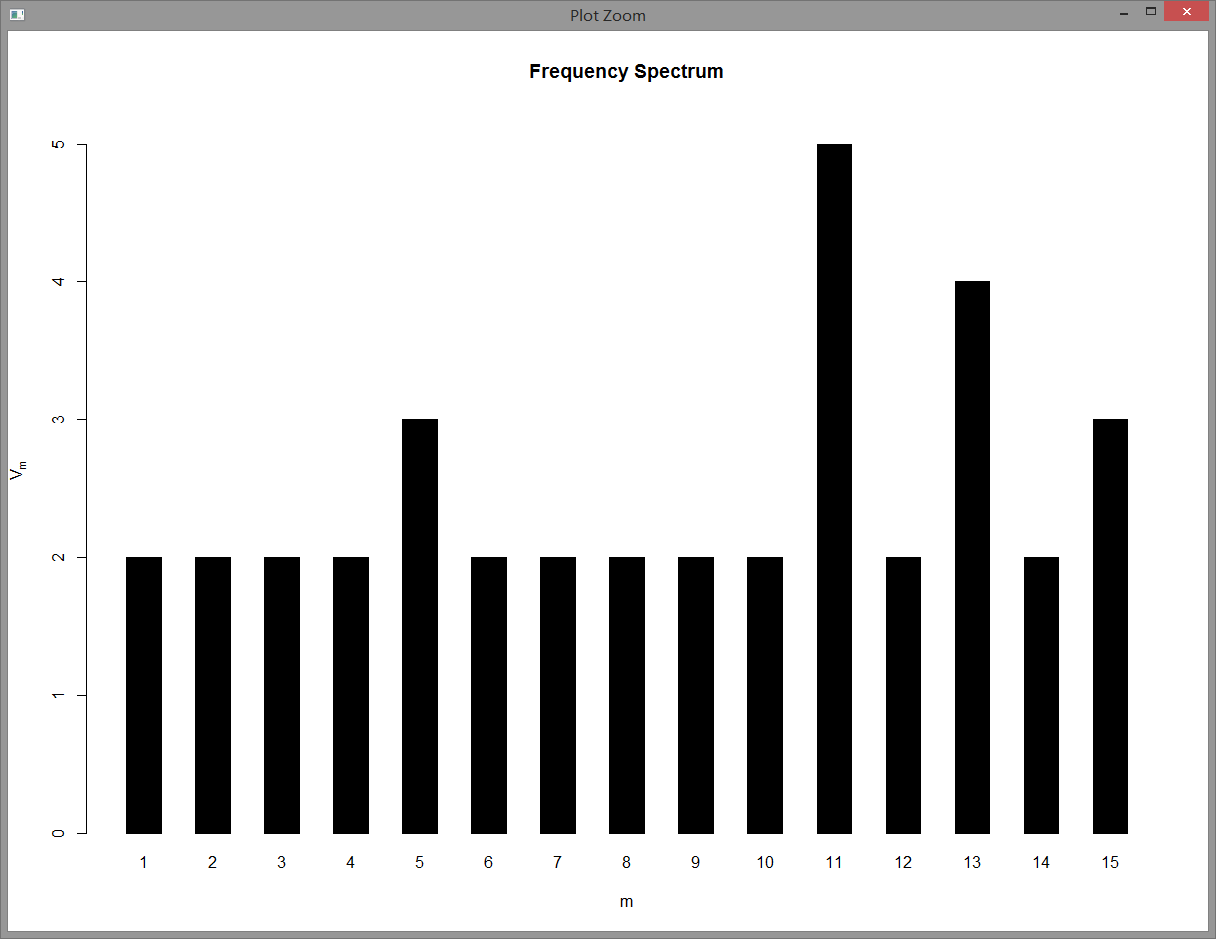
**acq[19]**



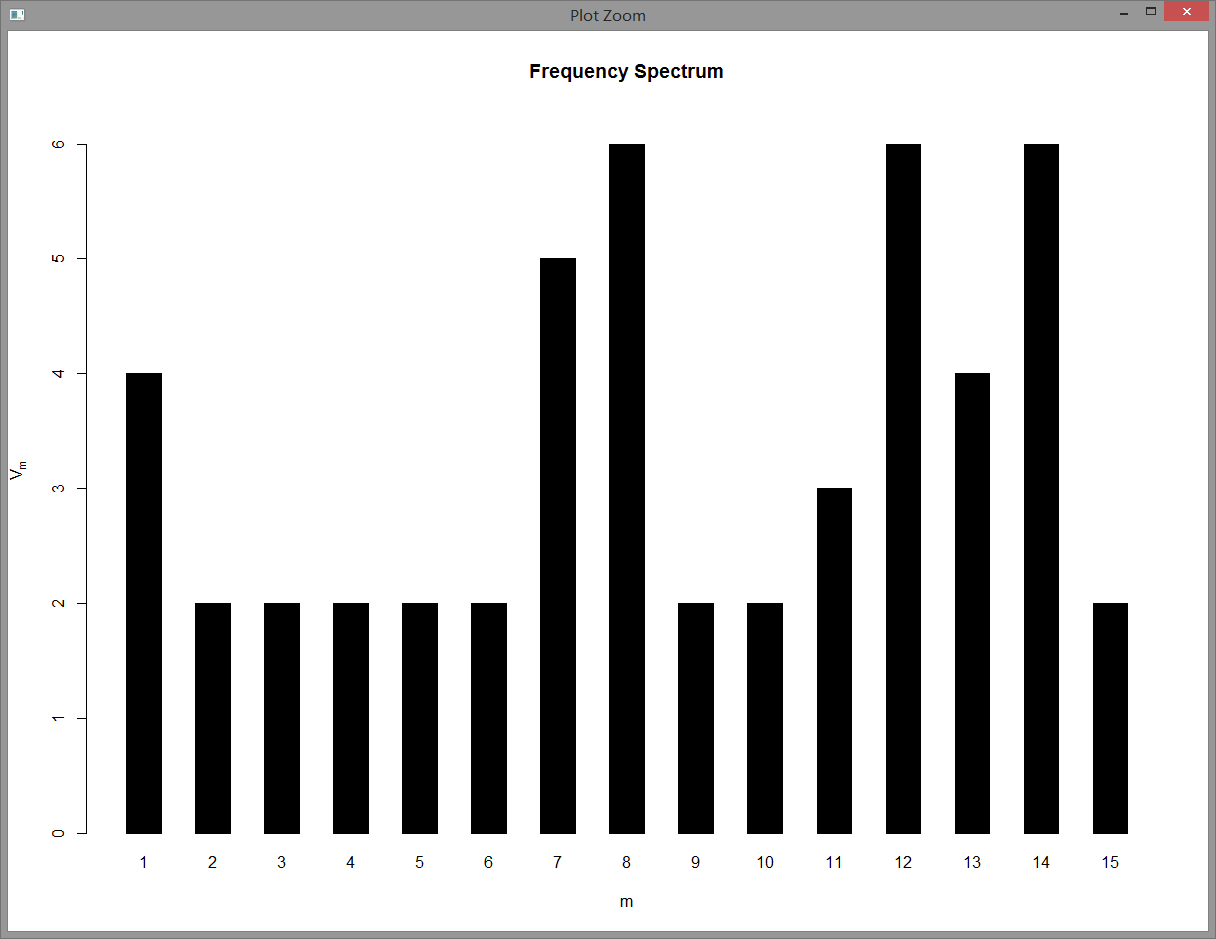
**acq[4]**



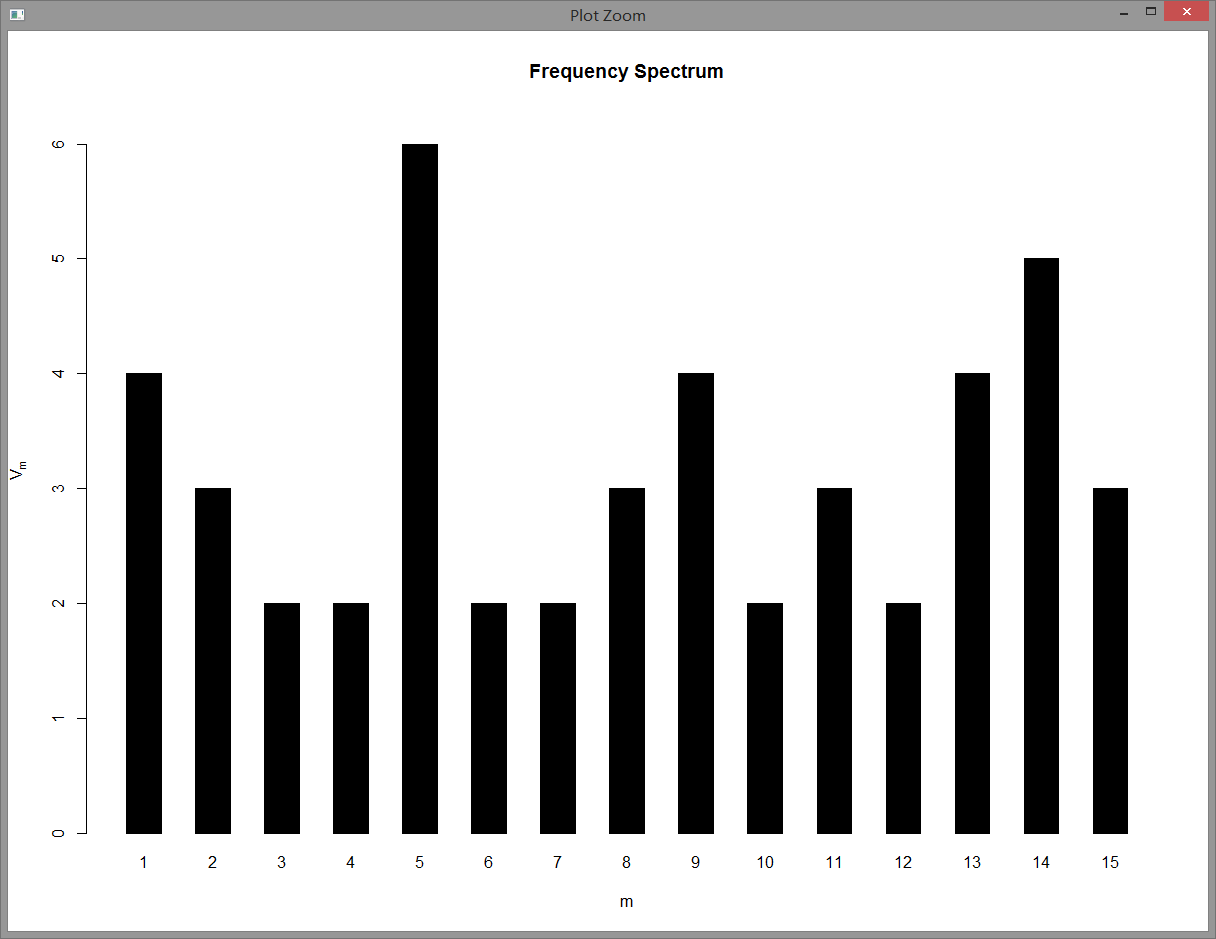
**acq[22]**



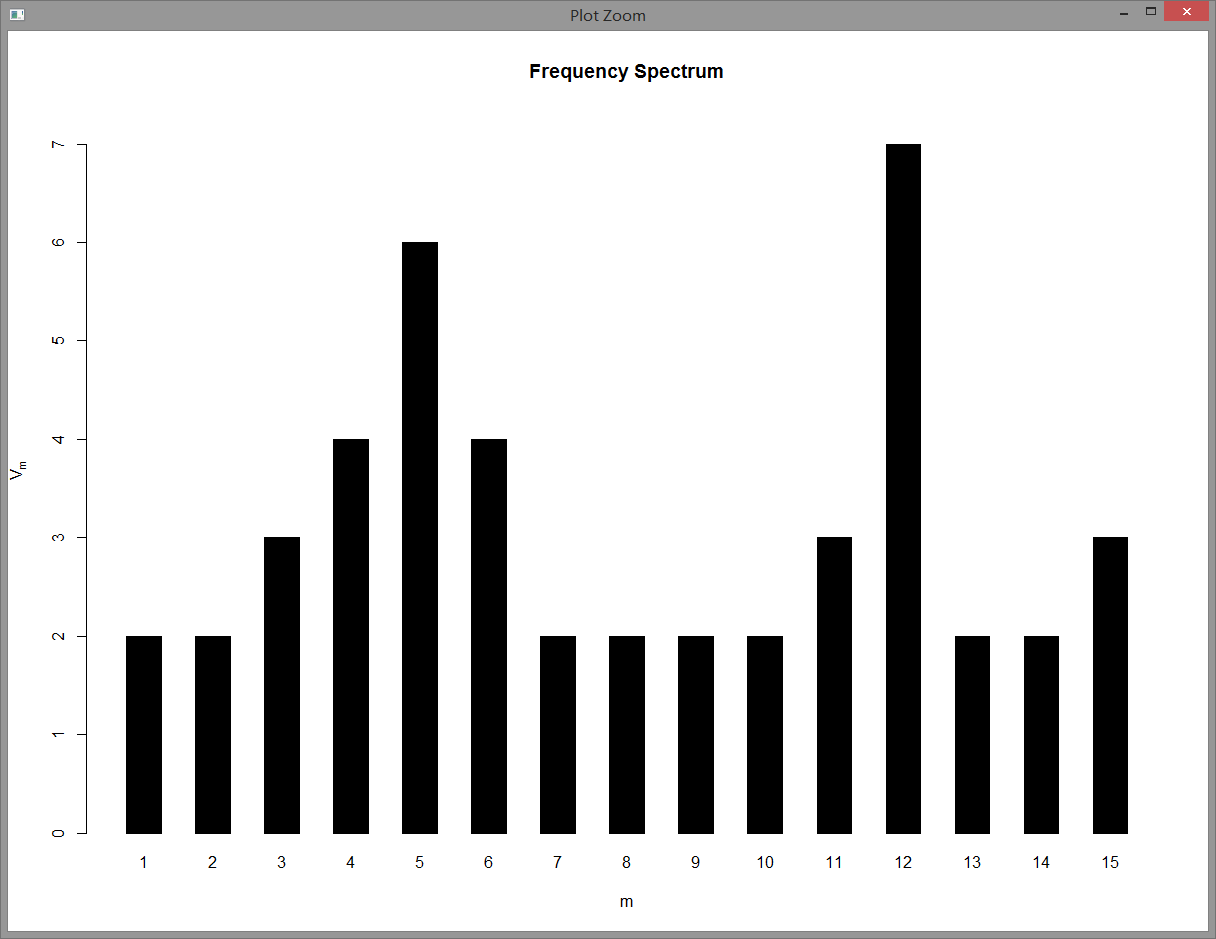
**acq[42]**



**acq[34]**



**acq[1]**



c. Write an R function to search through the documents to find a specific word or phrase. Print the document number, line number, and word index in the sentence. Demonstrate with three examples. Use words of 6 characters or more as your test cases.

#remove "\n"

> removen <-function(x) gsub("\n", " ", x);

> acqr <- tm\_map(acq, content\_transformer(removen));

#create a 10\*3 array to store the result of # of document, # of sentence, index of word

> e=array(dim = c(10,3));

> count=1;

> for (i in 1:50){

#store as sentences

a=acqr[i]

b=tokenize\_sentences(a[[1]]$content);

for(j in 1:length(b)){

c=b[j];

if(c!=""){

#split to get words

d=unlist(strsplit(c," "));

for(k in 1:length(d)){

#find the target words

if(d[k]=="warrants"){

e[count,1]=i;

e[count,2]=j;

e[count,3]=k;

count=count+1;

}

else{count=count}

}

}

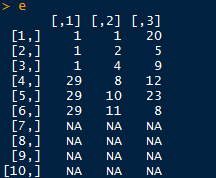
}

}

**Example 1:**

Search word: “warrants”

The result:



The result shows the word “warrants” showed 6 times in these locations:

1. document: acq[1], sentence: 1, word: 20

2. document: acq[1], sentence: 2, word: 5

3. document: acq[1], sentence: 4, word: 9

4. document: acq[29], sentence: 8, word: 12

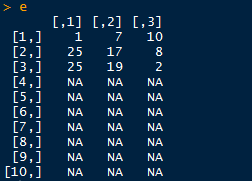
5. document: acq[29], sentence: 10, word: 23

6. document: acq[29], sentence: 11, word: 8

**Example 2:**

The searched word: “reorganization”

The result:



The result shows the word “reorganization” showed 3 times in these locations:

1. document: acq[1], sentence: 7, word: 10

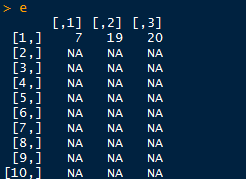
2. document: acq[25], sentence: 17, word: 8

3. document: acq[25], sentence: 19, word: 2

**Example 3:**

The searched word: “capitalization”

The result:



The result shows the word “reorganization” showed only once in the location:

1. document: acq[7], sentence: 19, word: 20

These results also match the outcome of question d in finding the longest words.

Discussion of results from the experiments

(1) The results are all correct. We can successfully clean the data, get the term frequency and plot it. The cluster dendrograms are right for each documents. The word cloud can properly present the distribution of the words.

(2) The 10 longest documents are correctly found by calculating the number of words.

(3) The longest words and sentences in each document are successfully found. We noticed that some documents have more than one longest word and sentence because they have same length.

(4) The table of the length of each sentence in each of the 10 documents is correctly established. We firstly store the text as sentence, and then calculate number of words in each sentence.

(5) After removing punctuation, the sentences in each document are successfully presented.

(6) The results of part of speech of words are achieved by applying the wordnet package. We use StartWithFilter as filter type, use ADJECTIVE as part of speech type. Some words has null result.

(7) Use spc function to get the frequency spectrum.

(8) Successfully write the function to search a specific word, return the document number, line number, and word index in the sentence. The result is stored in a 10\*3 array. And in our test case, we used “warrants” “reorganization” and “capitalization" as our searching words. The results are correct.