Project #3

Nisha Iyer, Rachel Jordan, Sam Dooley

April 30, 2016

We will perform analysis on a corpus of 50 documents from the acq dataset.

```r
data("acq")

#compilation of 50 news articles with additional meta information form the
#Reuters-21578 data se of topic acq. 13 documents
ACQ <- acq
```

Explore using functions from Lecture 7

We can reference information about the document with any of the following commands.

```r
#this tell us what information (metadata) about our documents.
# For example, how many chars are within each doc.
alldocs <- inspect(ACQ[1:2]) #just the first 2
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 2
##
## $`reut-00001.xml`
## <<PlainTextDocument>>
## Metadata:  15
## Content:  chars: 1287
##
## $`reut-00002.xml`
## <<PlainTextDocument>>
## Metadata:  15
## Content:  chars: 784
```

```r
# get the first document
text1 <- ACQ[[1]]

# get the id from the second document
id.2 <- ACQ[[1]]$meta$id
id.2 <- meta(ACQ[[1]], "id") #this is another way to reference
```

The command `meta` will return understandable information about the documents. It will tell you who wrote the article, when it was written, the heading of the article, its language, its origin, etc. This can be useful when searching for particular documents or languages.

This function tells us more information about the texts (all 50). For example, the maximal term length, non/sparse entries

```r
ACQdoc <-DocumentTermMatrix(ACQ)
ACQdoc
```

```
## <<DocumentTermMatrix (documents: 50, terms: 2103)>>
## Non-/sparse entries: 4135/101015
## Sparsity           : 96%
## Maximal term length: 21
## Weighting          : term frequency (tf)
```

The DocumentTermMatrix lists as its rows the documents in the corpus, and as it columns the words of the corpus. entries of this matrix are numbered values that indicate how many times given document (row) contains a given a word (column). This can be seen here:

```
inspect(ACQdoc[1:6,1:7])
```

```
## <<DocumentTermMatrix (documents: 6, terms: 7)>>
## Non-/sparse entries: 2/40
## Sparsity           : 95%
## Maximal term length: 11
## Weighting          : term frequency (tf)
##
##     Terms
## Docs -laval .125 .3322 "...that "(american) "any "bridge"
##   10      0    1     0        0           0    0        0
##   12      0    0     0        0           0    0        0
##   44      0    0     0        0           0    0        0
##   45      0    0     0        0           0    1        0
##   68      0    0     0        0           0    0        0
##   96      0    0     0        0           0    0        0
```

termFreq tells us more about an individual doc/text such as term freq within the document. We can also then rank the terms from most frequent to least.

```
test1tf <- as.data.frame(termFreq(text1))
#rank words most to least
rank_words <- as.data.frame(test1tf[order(test1tf, decreasing = T),])
head(rank_words)
```

```
##          test1tf[order(test1tf, decreasing = T), ]
## the                                             15
## said                                             7
## and                                              6
## computer                                         6
## its                                              5
## for                                              4
```

The tm_map and content_transformer transforms the data such as converting the terms to lower case. Converting text to lower case is helpful for matching words that can have different capitalization schemes. For instance, a word might appear at the beginning of the sentence, but it is important to be able to count that word as the same as if it were not capitalized.

```
# to lower case
ACQlow <- tm_map(ACQ, content_transformer(tolower))
```

We also remove characters that are Ebglish letters or spaces. This removes punctuation from the text that can cause issues later on. We note that this is not the ideal method for removing punctuation as hyphenated words like cross-sectional would be distorted to something that isn't a word. For the purposes here, this technique is okay.

```
#the next function removes anything other than English letters or spaces
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
ACQcl <- tm_map(ACQlow,content_transformer(removeNumPunct))
```

We also run into a problem if we wanted to analyze frequency of words. The problem is that some words are just obviously more frequent: the, a, of, etc. Thus, we create a class of words, called *stopwords* - which is a part of the tm and quanteda packages - which we wish to remove from the corpus.

```
#after converting the text to lower case, and removing punctionation
#we are going to remove stopwords (filler words such as a, an, the, etc.)
stopwords <- c(stopwords('english'))
ACQstop <- tm_map(ACQcl, removeWords, stopwords)
```

This creates an interesting point of analysis: *How much information or text do we lose when we remove stopwords?*

```
#here we can look at the first two text docs and see how the word count (char) differs
inspect(ACQ[1])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 1
##
## $`reut-00001.xml`
## <<PlainTextDocument>>
## Metadata:  15
## Content:  chars: 1287
```

```
inspect(ACQstop[1])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 1
##
## $`reut-00001.xml`
## <<PlainTextDocument>>
## Metadata:  15
## Content:  chars: 1030
```

We see that the first document of ACQ drops from 1287 characters to 1030 characters. This means that this document had abou at 20% reduction in the number of characters. We see that this is a pretty stable reduction across this corpus.

Now that we have removed the document's punctuation and stopwords, we put that corpus back into a `DocumentTermMatrix`.

```
#now we are putting the terms without punctuation and stopwords into a matrix
ACQdm2 <- DocumentTermMatrix(ACQstop, control= list(wordLenghts = c(1,Inf)))
ACQdm2
```

```
## <<DocumentTermMatrix (documents: 50, terms: 1502)>>
## Non-/sparse entries: 2998/72102
## Sparsity           : 96%
## Maximal term length: 20
## Weighting          : term frequency (tf)
```

We also use the function `findFreqTerms` to look through the `DocumentTermMatrix` to find those words that were used a certain number of times or were used in a range of times.

```
#find terms with a frequency between 15 and 18
freq.terms <- findFreqTerms(ACQdm2, lowfreq=15, highfreq = 18)
freq.terms
```

```
## [1] "acquire"  "bank"     "business" "one"      "rmj"      "value"
## [7] "viacom"
```

We also have a function that will find words in your corpus - really your `DocumentTermMatrix` - and determine which of those words are Associates to another word above a given correlation score. We note that this is a correlation based of of how words are used in the `DocumentTermMatrix`, not similarity of the string like a Levenshtein distance or something.

```
#the Assocs function finds associations with terms, such as states or year
findAssocs(ACQdm2, "states", 0.6)
```

```
## $states
##         areas     arranging     assurance    bankruptcy        bodies
##          0.70          0.70          0.70          0.70          0.70
##      charters     continues      contract         court       crowley
##          0.70          0.70          0.70          0.70          0.70
##       delayed     equitable exchangeable         final      fraction
##          0.70          0.70          0.70          0.70          0.70
##     holdingss       include      includes          life         lines
##          0.70          0.70          0.70          0.70          0.70
##     mariotime        mclean       present       raising      revision
##          0.70          0.70          0.70          0.70          0.70
##       society     transport          used        united           mcv
##          0.70          0.70          0.70          0.69          0.66
##        raised    amusements      transfer      national
##          0.63          0.62          0.62          0.60
```

We thus conclude that the different functions allow us to break down the different text documents we were able to see how many stopwords and punctuation was included in the total character count of the texts the term frequencies allowed us insight into the top frequented words in the text the functions provided a lot of insight into the general documents, text, and words used in the texts

Find the 10 longest documents (in number of words)

```
#using quanteda for the next few questions
data("acq")
mycorpus <- corpus(acq)
summary_acq <- as.data.frame(summary(mycorpus))

#10 longest documents in the corpus
sort_top10 <- summary_acq %>% arrange(desc(Tokens))
top_10_docs <- subset(sort_top10, select=c(id, heading))[1:10,]
top10 <- top_10_docs[,1]
topdocs <- mycorpus[mycorpus$documents$id %in% top10]
```

We see from the above that the document IDs in the from the corpus' metadata are listed below. The order is in decreasing order by number of words.

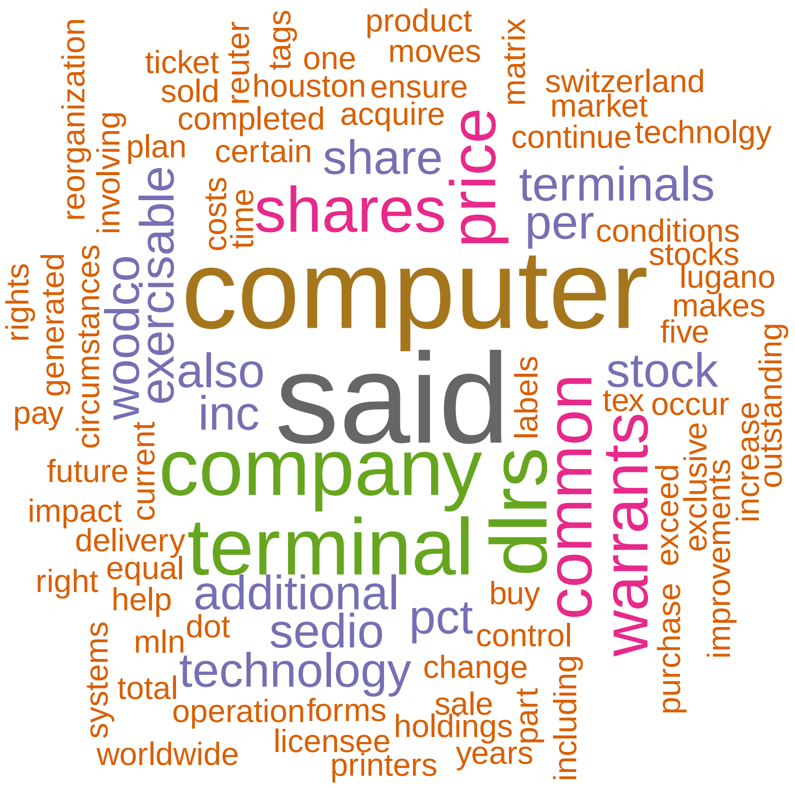
```
top10
```

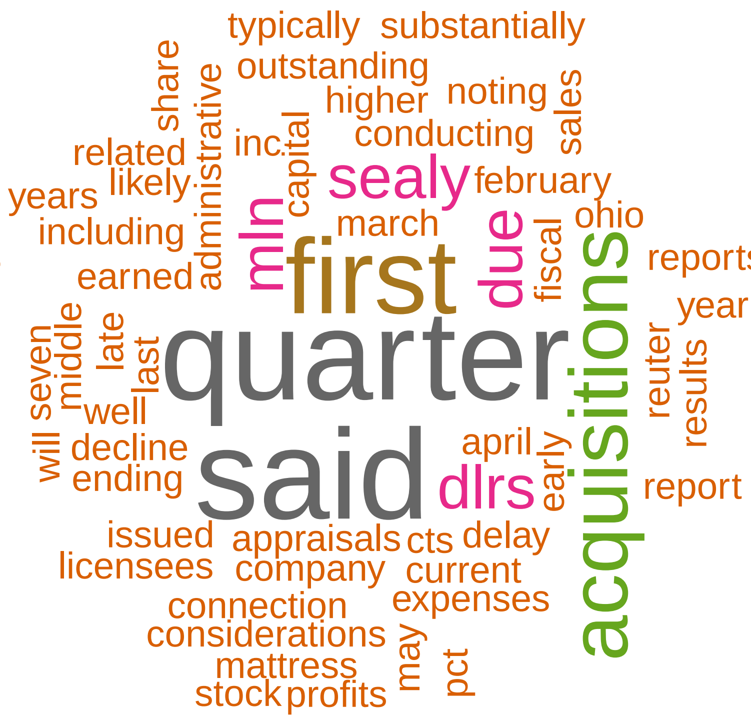
```
##  [1] "110" "362" "372" "496" "302" "45"  "331" "448" "393" "10"
```

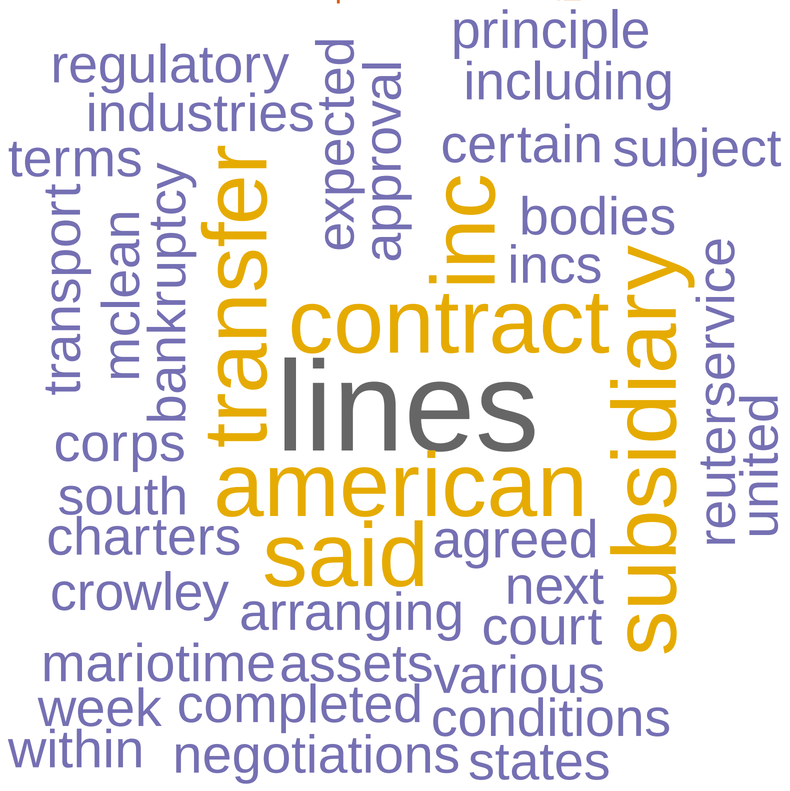
For each document work through the examples given in Lecture 7 to display the dendrogram and the WordCloud

```
#top 10 dendogram, 1 for each of the top 10 documents
top10.dendogram <- function(tdm2,doc)
{
  acq.mat <- as.matrix(tdm2)
  acq.mat <- as.data.frame(acq.mat)
  acq.mat <- acq.mat[,top10]
  acq.mat <- as.matrix(acq.mat)
  distMatrix <- dist(scale(acq.mat[,doc]))
  fit <- hclust(distMatrix, method = "ward.D2")
```

```r
  print(plot(fit,main = "Dendogram"))
}


#word cloud for top 10
wordcloud.func <- function(ACQstop, doc)
{
  dtm <- TermDocumentMatrix(ACQstop)
  v <- as.matrix(dtm[,doc])
  set.seed(1234)
  wordcloud(words = rownames(v), freq = v, min.freq = 1,
            max.words=200, random.order=FALSE, rot.per=0.35,
            colors=brewer.pal(8, "Dark2"))
}
for (i in 1:10){
  wordcloud.func(ACQstop,i)
}
```





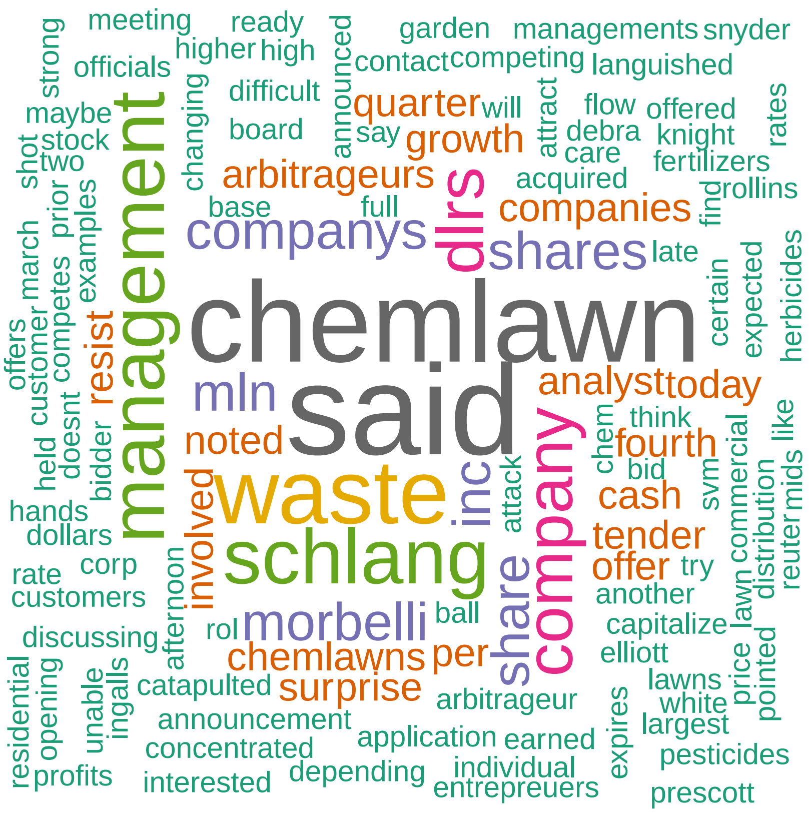


```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : overthecounter could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : regularly could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : removal could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : rosemarie could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : scheduled could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : servicemaster could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : sevice could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : slowed could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : spokeswoman could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : statement could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : street could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : strohmaier could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : system could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : thousands could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : times could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : totally could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : trading could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : turben could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : valuable could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : wall could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : want could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : wants could not be fit on page. It will not be plotted.
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : wastes could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : whether could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : wnx could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : year could not be fit on page. It will not be plotted.
```



```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : petrochemical could not be fit on page. It will not be plotted.
```

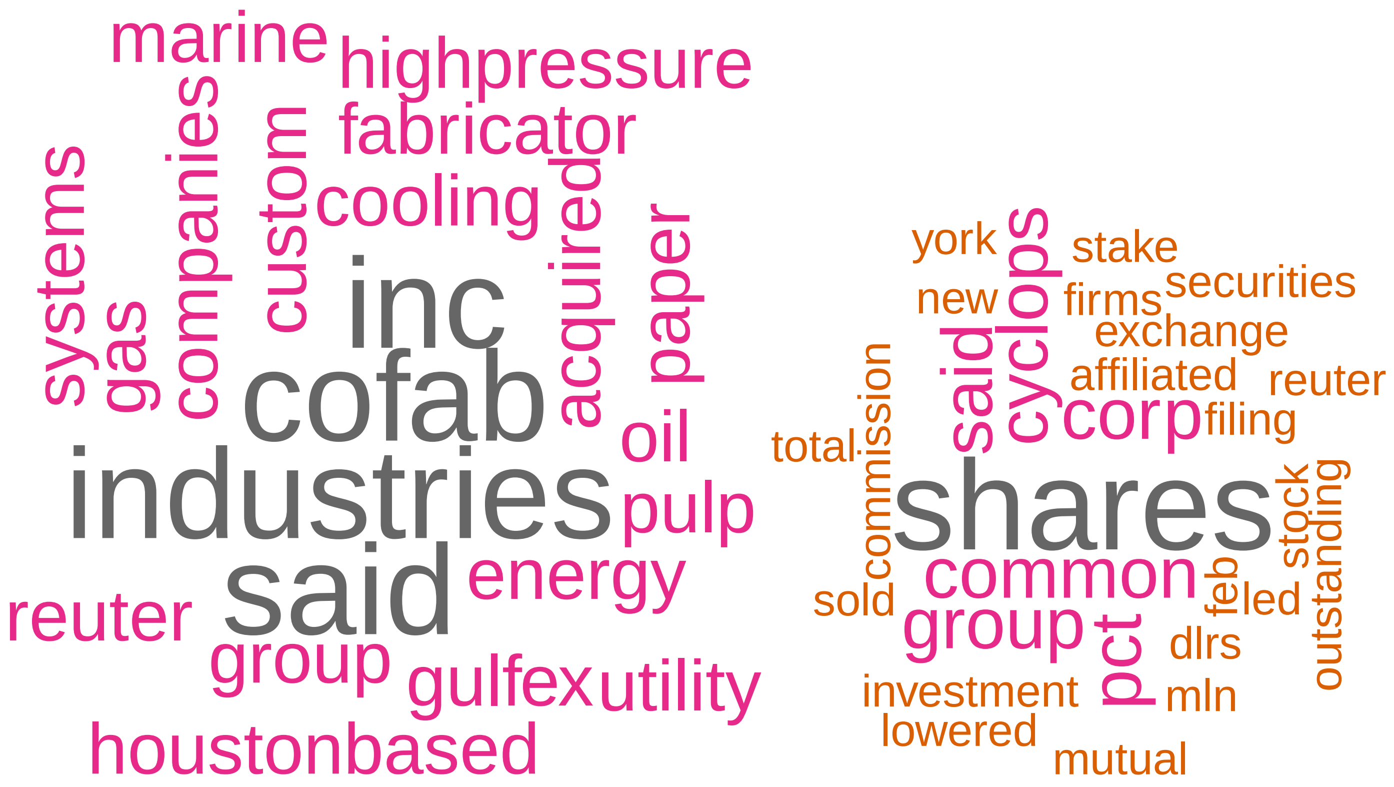
```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : lubricating could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : manufacture could not be fit on page. It will not be plotted.
```

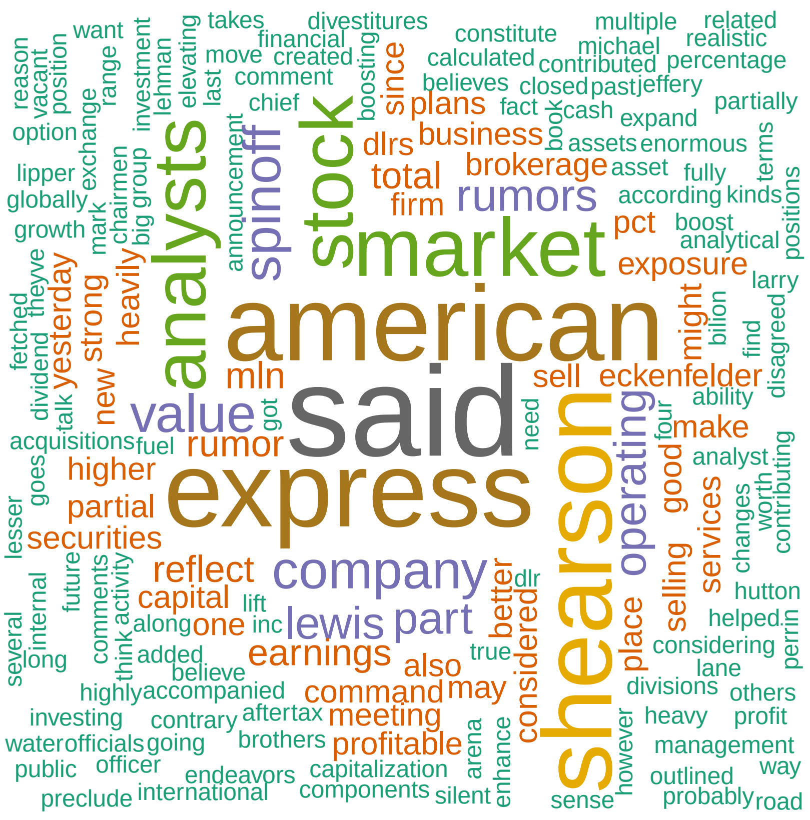
```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : process could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : specialized could not be fit on page. It will not be plotted.
```

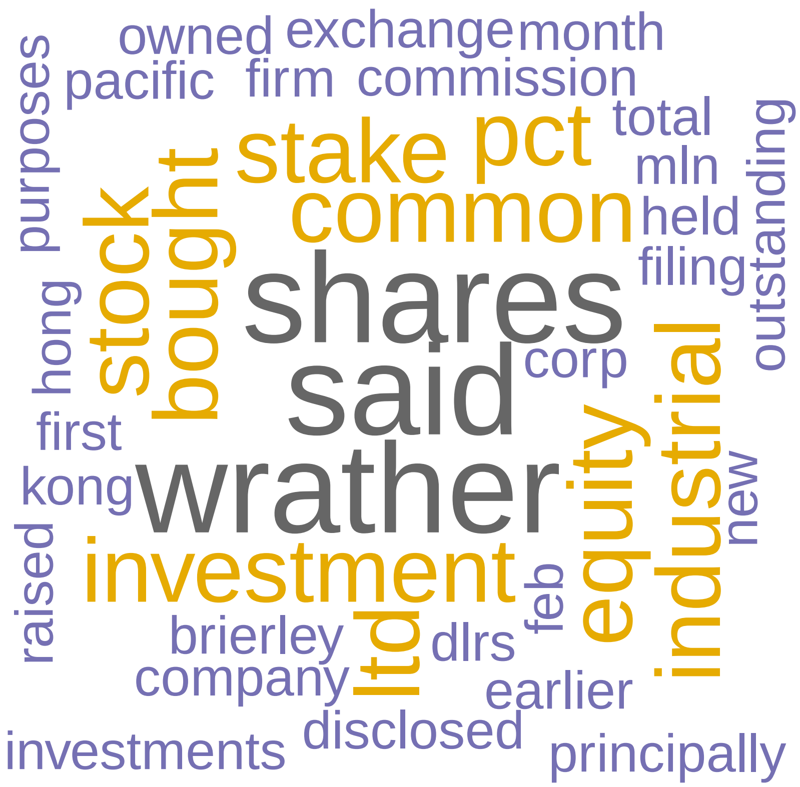
```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : vessels could not be fit on page. It will not be plotted.
```

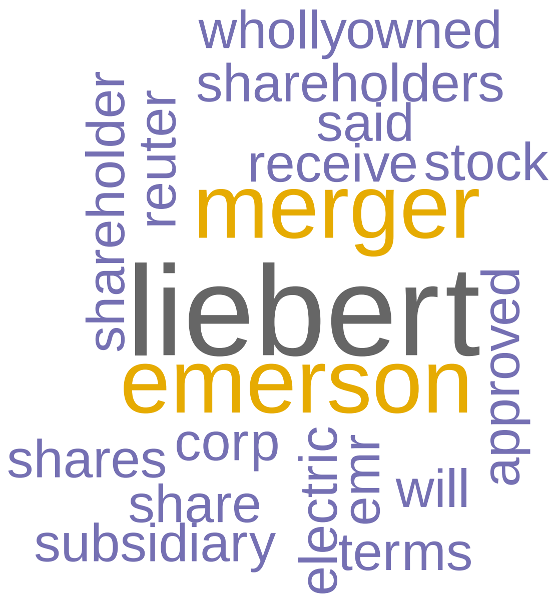


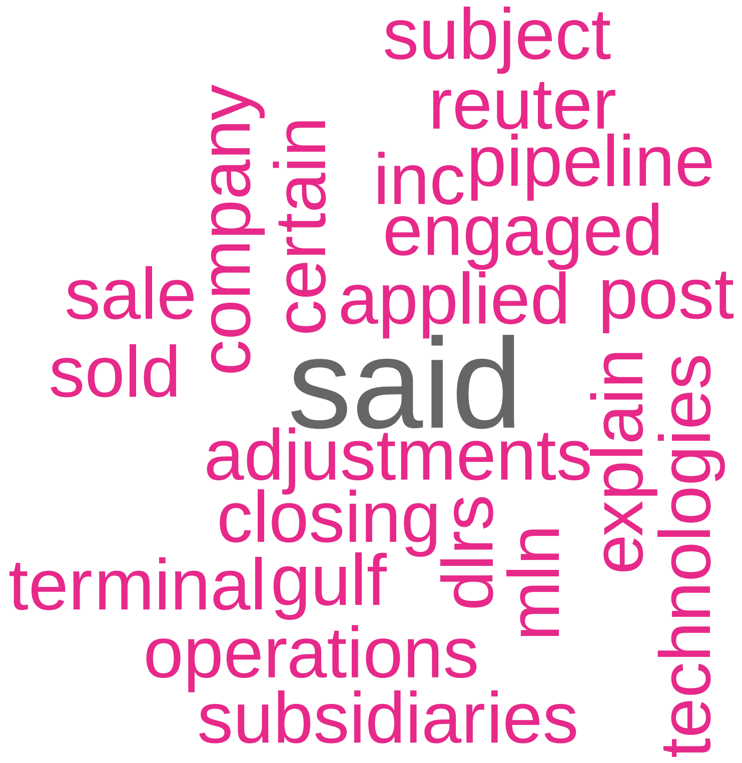
```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : performance could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : premium could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : president could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : prudentialbache could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : questioned could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : reduced could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : remained could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : reuter could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : shearsons could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : speculated could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : spinning could not be fit on page. It will not be plotted.
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : split could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : suggests could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : thereby could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : theyre could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : three could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : tuesday could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : undervalued could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : unhappy could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : unlikely could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : use could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : variation could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : volume could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : well could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : year could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : york could not be fit on page. It will not be plotted.
```



```
## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : publicly could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : reuter could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : securities could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = rownames(v), freq = v, min.freq = 1, max.words
## = 200, : zealand could not be fit on page. It will not be plotted.
```







Prior to removing punctuation find the longest word and longest sentence in each of 10 docs my corpus is before removing punctuation

The project helped us learn a lot about text analytics and key principals of analyzing unstructured text.We identified three key areas this project helped you learn about data science includes (1) the general approach to breaking down texts in R using Corpuses and tokens; (2) The exploratory analysis and derived insights that can be accomplish on a text documents through word counts, frequencies, associations, and character lengths; (3) we were able to learn how to apply data mining techniques to text analytics for deeper insights such as clustering (hierarchical and kmeans).