Final Assignment

Suchandra Mandal

12 April 2018

Finding out the Topic Compositions for the given corpus.

In this assignment, we have been provided with 7,142 short text documents in 19 folders and our task is to understand the topics that the texts talk upon. For this, we first read in the corpus using the VCorpus function.

Load the libraries

We first load the libraries which we will be using throughout this assignment.

```
library(tm)
```

```
## Loading required package: NLP
```

```
library(SnowballC)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(cluster)
library(HSAUR)
```

```
## Loading required package: tools
```

```
library(fpc)
library(skmeans)
```

Reading in the corpus

```
corpus <- VCorpus(DirSource("C:/Users/Suchandra/Documents/SEM2/Data Visualisations/ASSIGNMENT
S/Final Assignment/corpus_n_topics3", recursive = TRUE, encoding = "UTF-8"), readerControl =
 list(language = "eng"))
```

Cleaning the data

We see that the corpus has 7142 elements or short text documents in it. First, we do certain preprocessing steps before we perform any tf-idf on it so that the important terms will be used to form our term document matrix. By important terms I mean, words excluding stopwords and other letters or punctuations.We clean the data by removing all special characters such as /,,@, etc to blank space using gsub; removal of stopwords since they are the most frequent words in any document and including them in our data will get us wrong results as they will have the highest frequency, punctuations is done next so that they do not feature in our visualisations. Also, all words are changed to one case, lower I have taken here. Also, the words in the corpus have been stemmed such that, for instance different tenses of a word or similar words such as computer, computing etc become comput.

```
# Calling the content_transformer function to replace different patterns with blank # space
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))

# replacing the special characters with blanks
corpus <- tm_map(corpus, toSpace, "/")
corpus <- tm_map(corpus, toSpace, "/.")
corpus <- tm_map(corpus, toSpace, "@")
corpus <- tm_map(corpus, toSpace, "\\|")

# Convert the text to lower case
corpus <- tm_map(corpus, content_transformer(tolower))

# remove all stopwords from the corpus
corpus <- tm_map(corpus, removeWords, stopwords("english"))

# remove punctuations
corpus <- tm_map(corpus, removePunctuation)

# Remove numbers and letters
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, c(letters))

# stem the words in the corpus
corpus <- tm_map(corpus, stemDocument)
```

Here, using the tm_map function, I have converted all words in document to lower case, removed punctuations, numbers and stemmed the words in the corpus.

Make a Document Term Matrix using Term Frequency

I have used tf-idf weighting to get term frequency of the words and multiplying that with the document frequency of terms to get a matrix of terms in our corpus sample. Next, I have removed sparse terms from the corpus object.The sparsity of the rows that we want to be removed can be given as input to the remove Sparse terms function. At first, I had tried removing rows with 20% sparsity to make the matrix dense but that resulted in most of the context giving words getting removed. Thus, have kept the limit quite high at .999. Also, rows which have no entries have been removed to make the matrix compact and reduce size.

```
# using tf-idf weighting scheme to form the document term matrix
#corpus.dtm.tf <- DocumentTermMatrix(corpus, control = list(weighting = weightTf))
corpus.dtm <- DocumentTermMatrix(corpus, control = list(weighting = function(x) weightTfIdf
(x, normalize = TRUE)))

corpus.dtm<-removeSparseTerms(corpus.dtm, 0.999)


corpus.dtm.mat <- corpus.dtm %>% as.matrix()


# remove any zero rows
corpus.dtm.mat <- corpus.dtm.mat[rowSums(corpus.dtm.mat^2) !=0,]
```

Since our Corpus is huge, I have taken a random sample of the corpus matrix. A 25% sample has been taken at random from the corpus matrix.

```
#Randomly sample a percentage of documents from the corpus - for efficiency purposes
percent = 25
sample_size = nrow(corpus.dtm.mat) * percent/100

corpus.dtm.mat.sample <- corpus.dtm.mat[sample(1:nrow(corpus.dtm.mat), sample_size, replace=F
ALSE),]
```

Using K-Means and Dendograms for the Corpus Sample to guess at the Number of clusters

Since the documents are all mixed and we need to find out clusters from the sample we have in order to understand the topic compositions of the documents, I have used both k-means and dendogram to get an understanding of the clusters of the terms in the corpus sample. Here, cosine has been used which is a similarity metric in the dendogram constructed and later has been converted to a distance metric by using the formulae since similarity is just the inverse of the distance between 2 terms.

For the K-means, I have executed K-means for clusters 1 to 30 and using elbow method, identified which K looks most likely to extract the various clusters.

```
#accumulator for cost results
cost_df <- data.frame()

#run kmeans for all clusters up to 100
for(i in 1:30){
  #Run kmeans for each level of i, allowing up to 100 iterations for convergence
  kmeans<- kmeans(x=corpus.dtm.mat.sample, centers=i, iter.max=100)

  #Combine cluster number and cost together, write to df
  cost_df<- rbind(cost_df, cbind(i, kmeans$tot.withinss))


}
names(cost_df) <- c("cluster", "cost")

#Calculate lm's for emphasis
lm(cost_df$cost[1:10] ~ cost_df$cluster[1:10])
```

```
##
## Call:
## lm(formula = cost_df$cost[1:10] ~ cost_df$cluster[1:10])
##
## Coefficients:
##           (Intercept)  cost_df$cluster[1:10]
##               960.056                -4.321
```

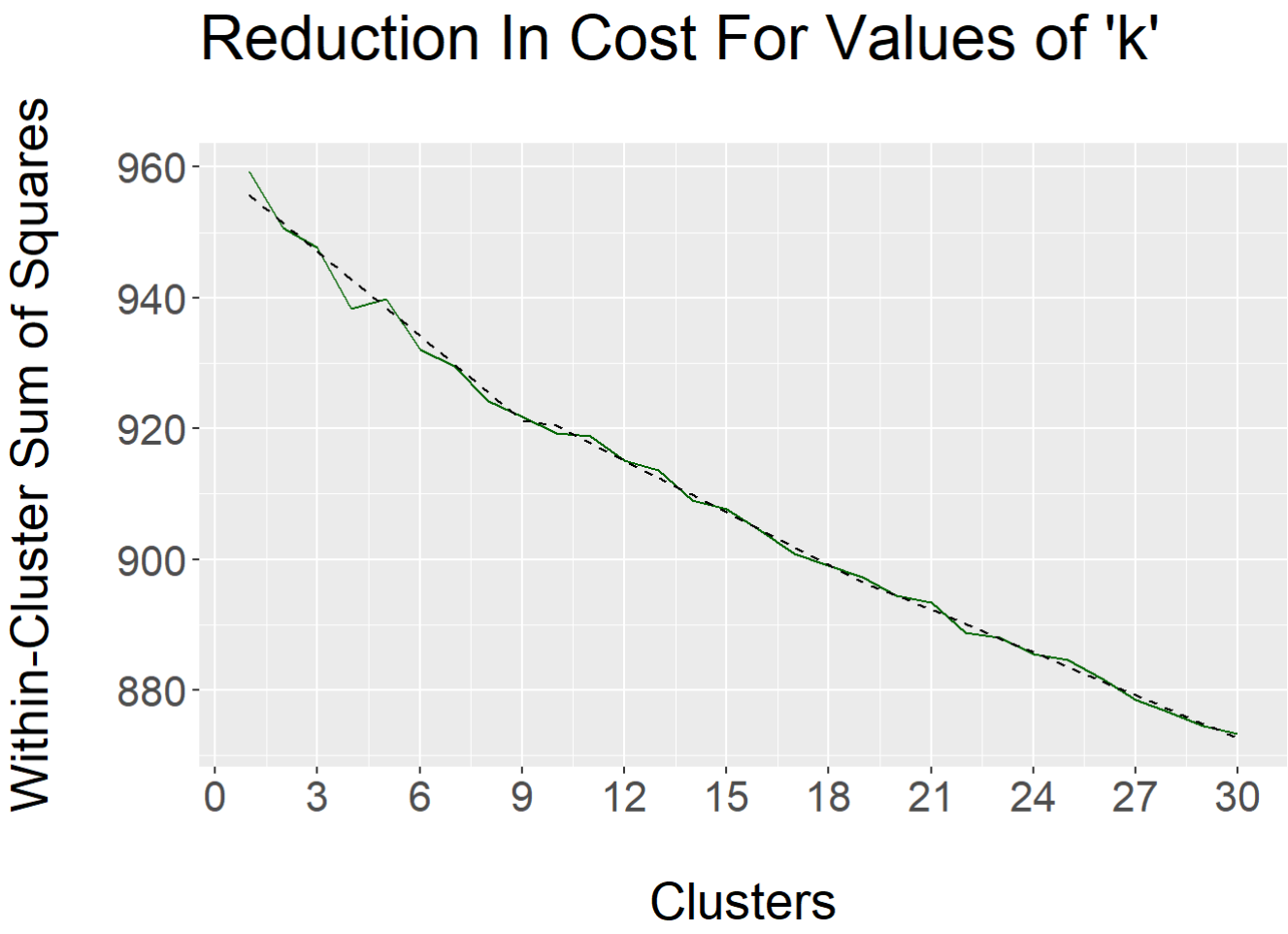
```
lm(cost_df$cost[10:19] ~ cost_df$cluster[10:19])
```

```
##
## Call:
## lm(formula = cost_df$cost[10:19] ~ cost_df$cluster[10:19])
##
## Coefficients:
##            (Intercept)  cost_df$cluster[10:19]
##                946.964                  -2.651
```

```
lm(cost_df$cost[20:30] ~ cost_df$cluster[20:30])
```

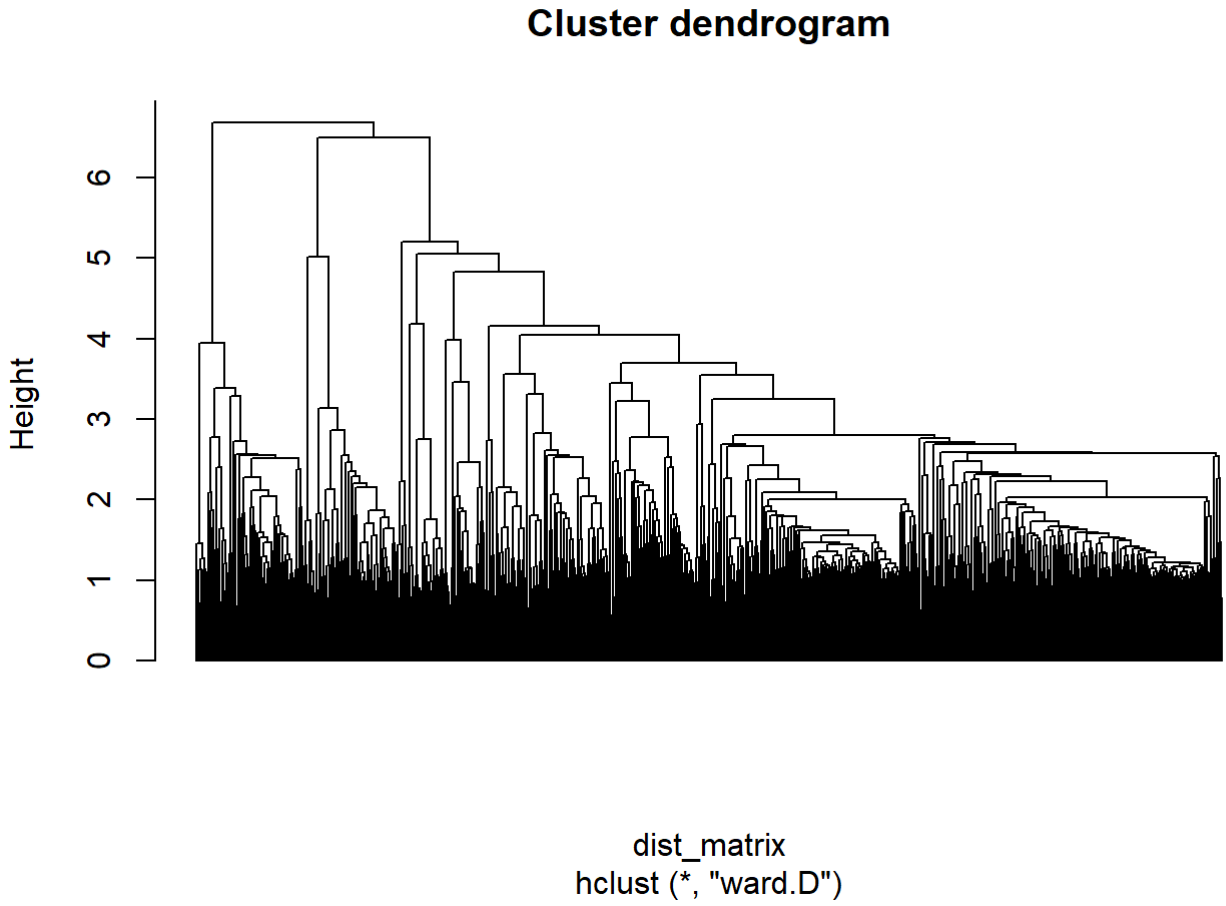
```
##
## Call:
## lm(formula = cost_df$cost[20:30] ~ cost_df$cluster[20:30])
##
## Coefficients:
##            (Intercept)  cost_df$cluster[20:30]
##                938.234                  -2.186
```

```
cost_df$fitted <- ifelse(cost_df$cluster <10, (960.056 - 4.321*cost_df$cluster),
                         ifelse(cost_df$cluster <20, (946.964 - 2.651*cost_df$cluster),
                         (938.234  - 2.186 *cost_df$cluster)))

#Cost plot
library(ggplot2)
ggplot(data=cost_df, aes(x=cluster, y=cost, group=1)) +
geom_line(colour = "darkgreen") +
theme(text = element_text(size=20)) +
ggtitle("Reduction In Cost For Values of 'k'\n") +
xlab("\nClusters") +
ylab("Within-Cluster Sum of Squares\n") +
scale_x_continuous(breaks=seq(from=0, to=30, by= 3)) +
geom_line(aes(y= fitted), linetype=2)
```



From the graph, we can see that the optimal number of clusters should be either 3 or 4. But since, the data may be hierarchical, we make our decision on optimal number of clusters based on dendograms. For the dendogram construction, we first define a similarity metric by using the cosine distance between two terms. The more similar 2 terms are, lesser will be the distance between them and thus, more is the similarity. We plot the dendogram for our corpus sample.

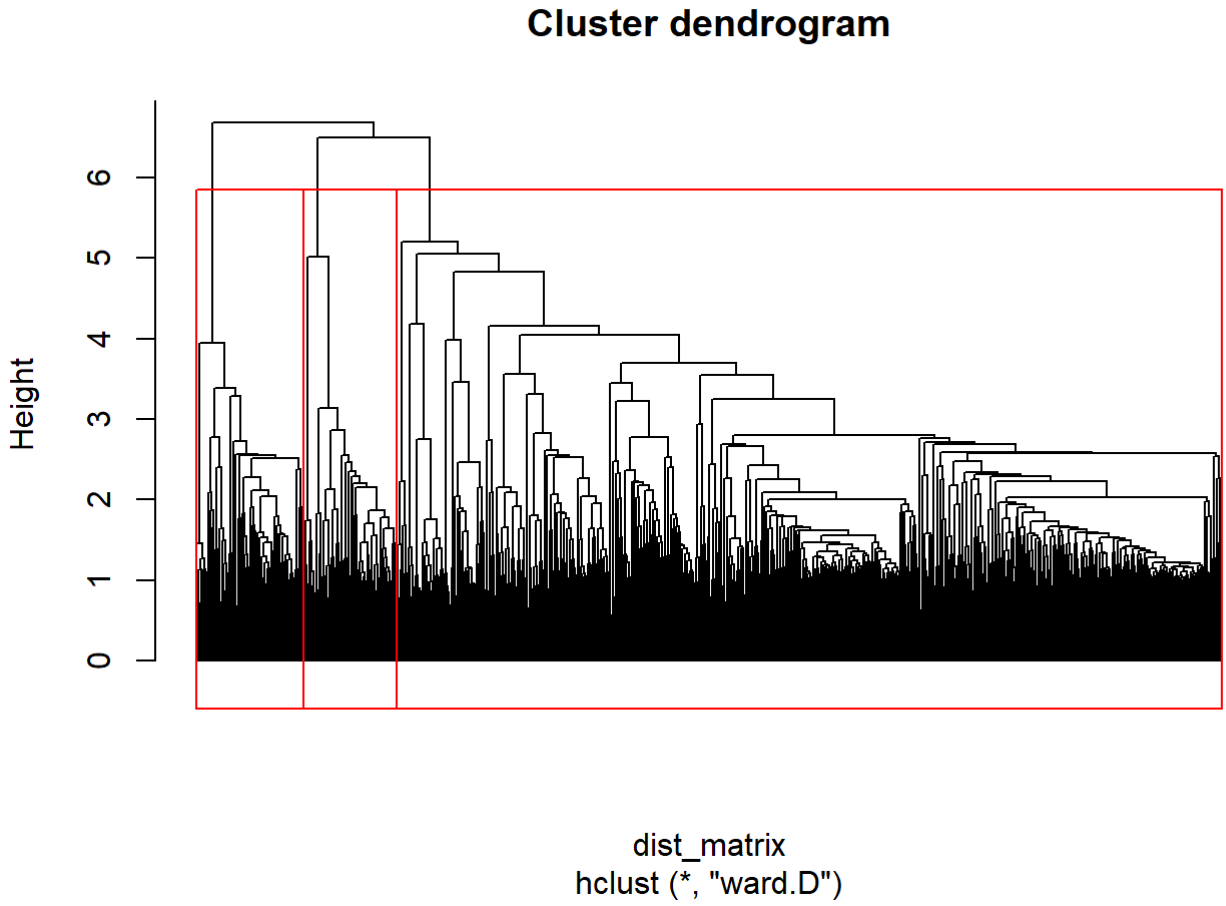
```
library(philentropy)

# from philentropy library. Slower than dist function, but handles cosine similarity
sim_matrix<-distance(corpus.dtm.mat.sample, method = "cosine")

# for readiblity (and debugging) put the doc names on the cols and rows
colnames(sim_matrix) <- rownames(corpus.dtm.mat.sample)
rownames(sim_matrix) <- rownames(corpus.dtm.mat.sample)


# to create a distance measure for hierarchical clustering
dist_matrix <- as.dist(1-sim_matrix)

# hierarchical clustering
corpus.dtm.sample.dend <- hclust(dist_matrix, method = "ward.D")

# plot the dendogram
# we hope to see some structure that reflects the finding of the kmeans algorithm

plot(corpus.dtm.sample.dend, hang= -1, labels = FALSE,  main = "Cluster dendrogram", sub = NU
LL, xlab = NULL, ylab = "Height")
```



From the dendogram, I could distinctly see 3 big clusters from the corpus sample chosen. Each of the clusters have 2 sub-clusters. And thereby, I will first check the words appearing in my cluster summary done later for 3 clusters first and then check the same for 4 and 5 clusters.Whichever cluster grouping seems the best at differentitating amongst the clusters will be chosen as the optimal one.

Let's then look at a summary of the clusters we have chosen. currently, I am going to view the corpus as 3 clusters. The clusters chosen have been highlighed in red

```
plot(corpus.dtm.sample.dend, hang= -1, labels = FALSE,  main = "Cluster dendrogram", sub = NU
LL, xlab = NULL, ylab = "Height")

# here rect.hclust creates  rectangles around the dendogram for k number of clusters
rect.hclust(corpus.dtm.sample.dend, k = 3, border = "red")
```



Summary of Clusters

Now let's look at summary of each cluster by looking at the top 25 words from each cluster and each cluster's size as in, the number of terms contained by each cluster. By top, I mean words that are most frequent in the clusters

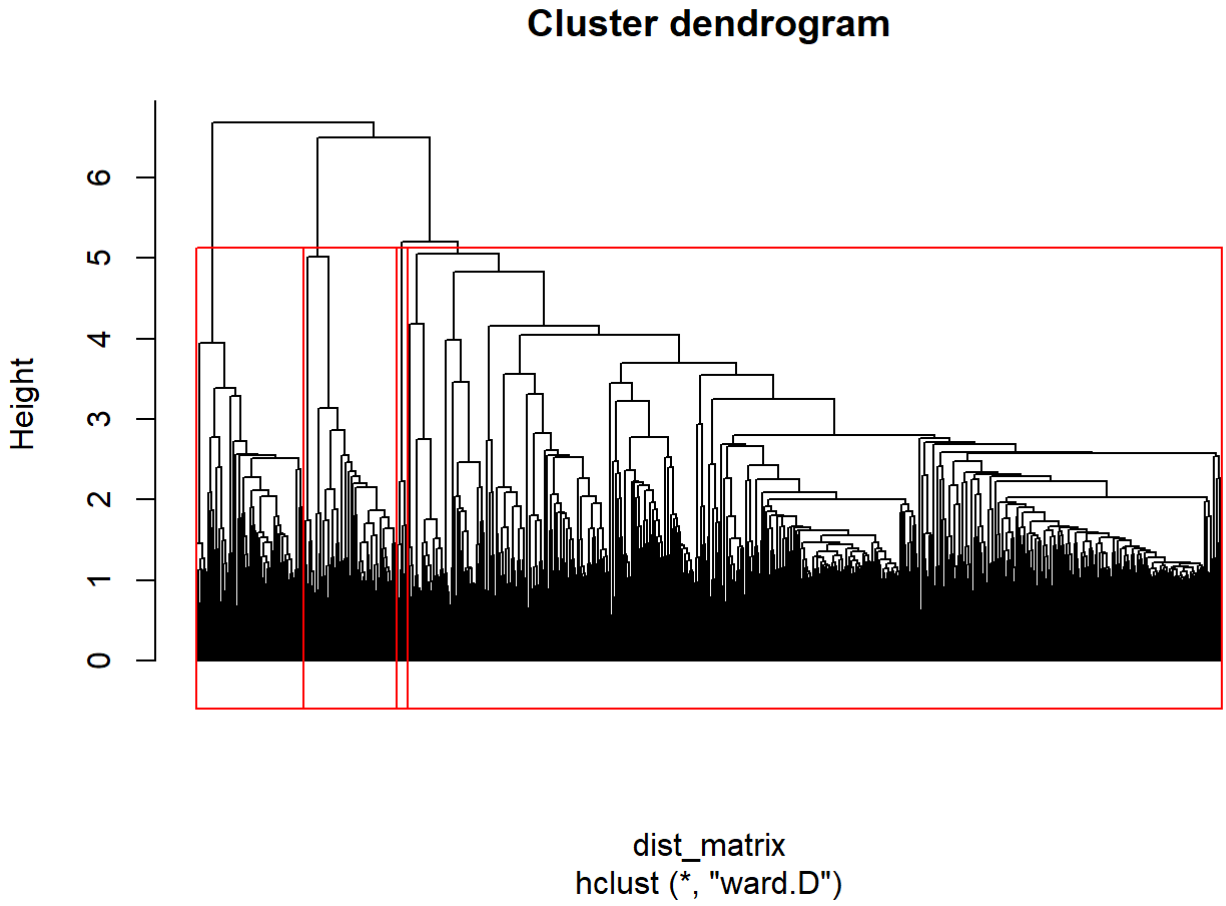
```
#It might be nice to get an idea of what's in each of these clusters. We can use the #probabi
lity difference method.

prob_terms <- colSums(corpus.dtm.mat.sample) / sum(corpus.dtm.mat.sample)
clusters <- cutree(corpus.dtm.sample.dend, k = 3)
clus_terms <- lapply(unique(clusters), function(x){
  rows <- corpus.dtm.mat.sample[ clusters == x , ]

  # for memory's sake, drop all words that don't appear in the cluster
  rows <- rows[ , colSums(rows) > 0 ]

  colSums(rows) / sum(rows) - prob_terms[ colnames(rows) ]
})

#
# create a summary table of the top 5 words defining each cluster
cluster_summary <- data.frame(cluster = unique(clusters),
                              size = as.numeric(table(clusters)),
                              top_words = sapply(clus_terms, function(d){
                                paste(
                                  names(d)[ order(d, decreasing = TRUE) ][ 1:25 ],
                                  collapse = ", ")
                              }),
                              stringsAsFactors = FALSE)
cluster_summary
```

```
##   cluster size
## 1       1 1436
## 2       2  187
## 3       3  162
##

            top_words
## 1                              window, use, file, program, system, car, comput, font, mac,
thank, card, drive, anyon, bike, problem, dos, monitor, help, imag, phone, distribut, bit, so
ftwar, email, inform
## 2                          game, team, espn, pitcher, player, hockey, fan, play, basebal,
don, pitch, mat, boston, pit, hit, spotcoloradoedu, robinson, leaf, win, franjion, bat, shar
k, sox, playoff, leagu
## 3 homosexu, god, christian, sin, cramer, christ, church, jesus, mari, gay, crep, sandvik,
bibl, clayton, athosrutgersedu, vers, scriptur, faith, doctrin, heaven, canon, interpret, opt
ilinkcom, believ, men
```

From the cluster summary, we can see that cluster 1 is related to computers; cluster 2 is quite well defined and has terms mostly related to games, cluster 3 seems to have topic related to religion with words such as faith, atheist, believe, homosexual etc

Now, to check if splitting our corpus sample to 4 clusters yeilds better cluster segregation and recognition of topics in cluster 1. We repeat the above code with k=4 to check the cluster summary for top 25 terms

```
plot(corpus.dtm.sample.dend, hang= -1, labels = FALSE,  main = "Cluster dendrogram", sub = NU
LL, xlab = NULL, ylab = "Height")

# here rect.hclust creates  rectangles around the dendogram for k number of clusters
rect.hclust(corpus.dtm.sample.dend, k = 4, border = "red")
```




```
prob_terms <- colSums(corpus.dtm.mat.sample) / sum(corpus.dtm.mat.sample)
clusters <- cutree(corpus.dtm.sample.dend, k = 4)
clus_terms <- lapply(unique(clusters), function(x){
  rows <- corpus.dtm.mat.sample[ clusters == x , ]

  # for memory's sake, drop all words that don't appear in the cluster
  rows <- rows[ , colSums(rows) > 0 ]

  colSums(rows) / sum(rows) - prob_terms[ colnames(rows) ]
})

#
# create a summary table of the top 5 words defining each cluster
cluster_summary <- data.frame(cluster = unique(clusters),
                              size = as.numeric(table(clusters)),
                              top_words = sapply(clus_terms, function(d){
                                paste(
                                  names(d)[ order(d, decreasing = TRUE) ][ 1:25 ],
                                  collapse = ", ")
                              }),
                              stringsAsFactors = FALSE)
cluster_summary
```

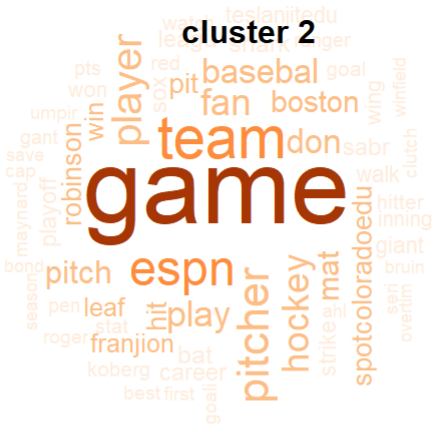
```
##   cluster size
## 1       1 1417
## 2       2  187
## 3       3  162
## 4       4   19
##

            top_words
## 1                                      window, program, system, file, comput, car, font,
use, mac, thank, card, drive, anyon, bike, help, dos, imag, email, bit, softwar, problem, dis
k, monitor, work, dod
## 2                           game, team, espn, pitcher, player, hockey, fan, play, basebal,
don, pitch, mat, boston, pit, hit, spotcoloradoedu, robinson, leaf, win, franjion, bat, shar
k, sox, playoff, leagu
## 3 homosexu, god, christian, sin, cramer, christ, church, jesus, mari, gay, crep, sandvik,
bibl, clayton, athosrutgersedu, vers, scriptur, faith, doctrin, heaven, canon, interpret, opt
ilinkcom, believ, men
## 4            sternlight, clipper, tap, david, key, strnlght, netcomcom, phone, nixon, omis
s, dsi, uscrpac, fed, ensur, accuraci, care, crypto, crook, taken, error, except, keyescrow,
serial, convers, chip
```

On splitting the corpus into 4 clusters, we see that cluster 1 is now completely related to computers, whereas cluster 2 is games related, cluster 3 is religion related but cluster 4 still seems mixed in and the topics in it seem related to security in computers mostly but since that is a subset of computers itself, I have taken 3 cluster groups as my optimal clusters.

Visualise the cluster compositions in word cloud

Once, I have fixed the number of topics I think are prsent in the corpus sample, let’s visualise the data using word clouds. Word clouds are extremely useful at getting to know at one the words in each of the cluster groups along with its frequency. The frequency being a function of size of the term in the word cloud gives us a fairly quick and comphrehensive idea about the composition of the terms in each of the cluster topics.

```
# plot a word cloud of one cluster as an example

corpus.dtm.sample.dend.cut = cutree(corpus.dtm.sample.dend, k =3)
# create a data frame from the cut
corpus.dtm.sample.dend.cut <- as.data.frame(corpus.dtm.sample.dend.cut)

#add a meaningful column namane
colnames(corpus.dtm.sample.dend.cut) = c("cluster")
#number of clusters at the cut
m <- length(unique(corpus.dtm.sample.dend.cut$cluster))

par(mfrow=c(2,2))
for(i in 1:m){
wordcloud::wordcloud(words = names(clus_terms[[ i ]]),
                     freq = clus_terms[[ i ]],
                     scale=c(4,.2),
                     max.words = 60,
                     random.order = FALSE,
                     rot.per=0.35,
                     colors=c('#feedde','#fdbe85','#fd8d3c','#e6550d','#a63603'))

title(paste("cluster", i))

}
```



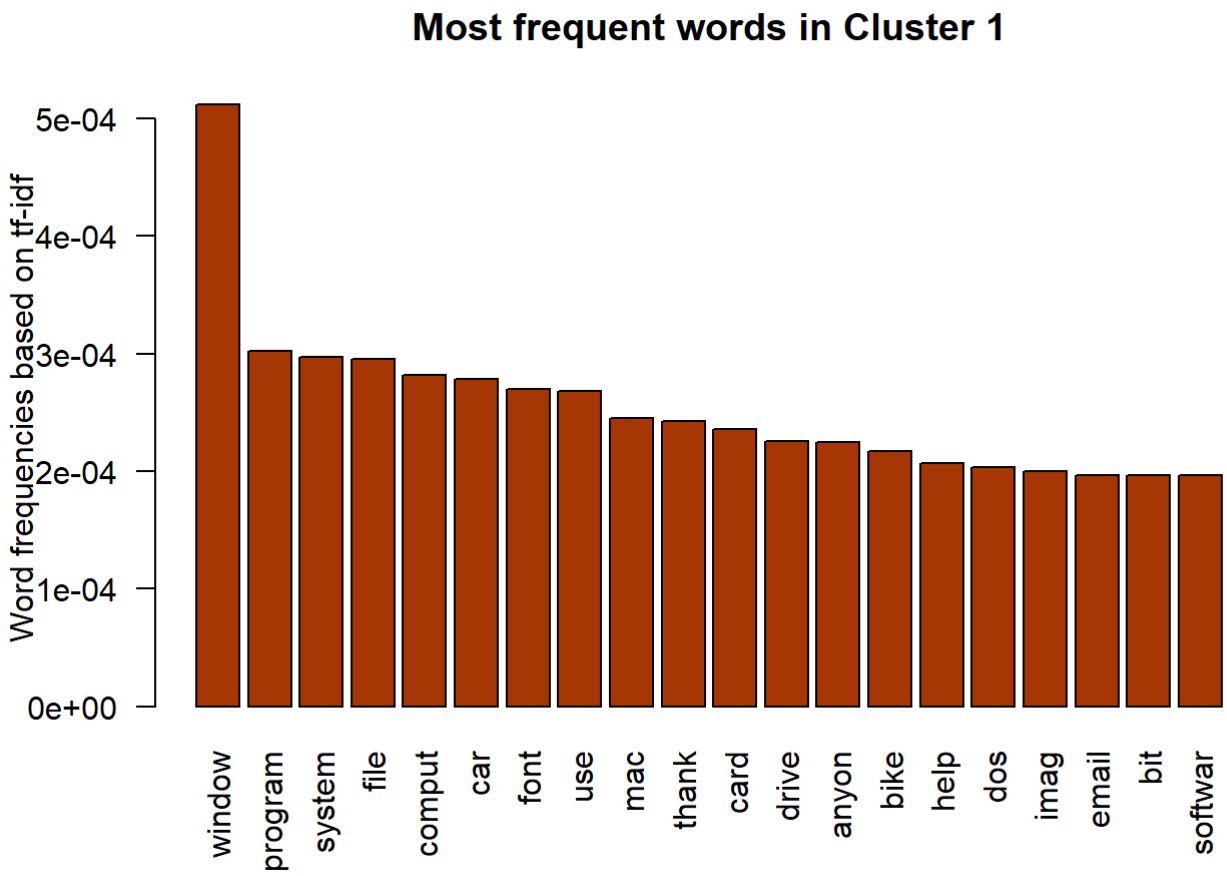
Plotting barplots to get an idea of the topics compositions in the 3 clusters chosen

From the word cloud, we see the most frequent words in each of the clusters whose topics are computers, game and religion.

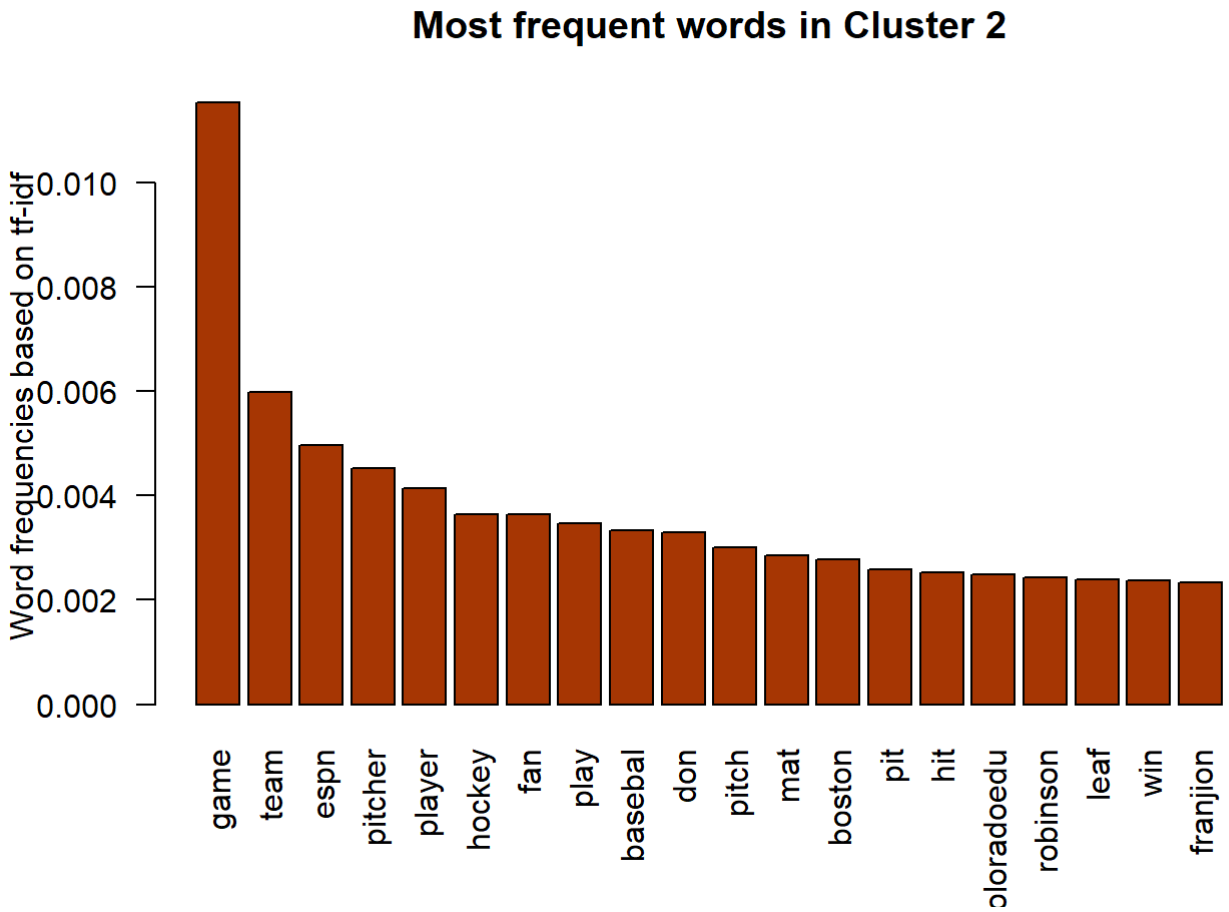
From the word cloud, we get an idea of which words or terms are most representative of my clusters but it does not tell us about the exact compositions of the clusters. Thus, next we look at bar charts to understand the exact compositions in our cluster topics.

Since I have used tf-idf as my weighting scheme to generate the document term matrix, the frequencies represnted in barplots are smaller than actual frequencies

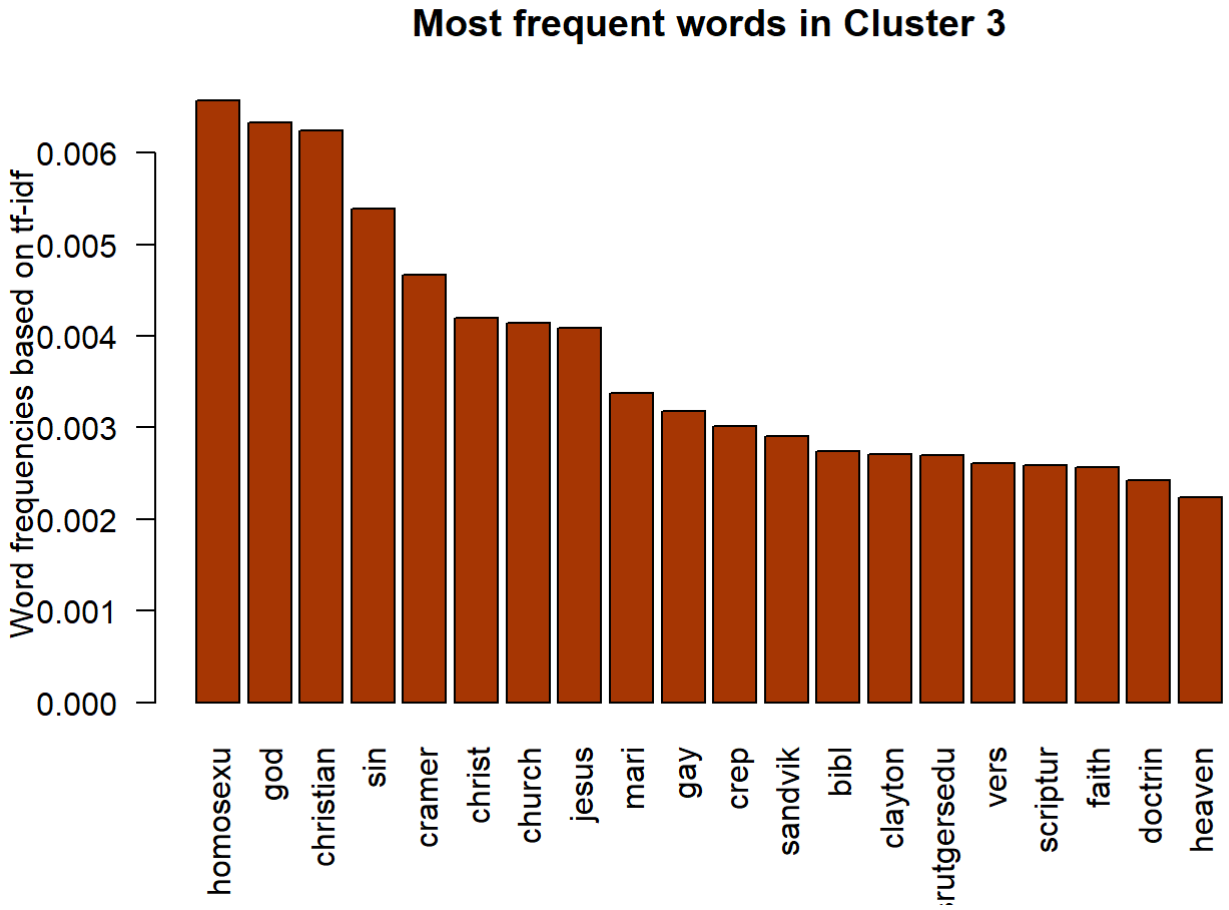
```
barplot(clus_terms[[ 1 ]][ order(clus_terms[[1]], decreasing = TRUE) ][ 1:20 ], las = 2, name
s.arg = names(clus_terms[[1]][ order(clus_terms[[1]], decreasing = TRUE) ][ 1:20]),col = '#a6
3603', main ="Most frequent words in Cluster 1",
        ylab = "Word frequencies based on tf-idf")
```




```
barplot(clus_terms[[ 2 ]][ order(clus_terms[[2]], decreasing = TRUE) ][ 1:20 ], las = 2, name
s.arg = names(clus_terms[[2]][ order(clus_terms[[2]], decreasing = TRUE) ][ 1:20]),col ='#a63
603', main ="Most frequent words in Cluster 2",
        ylab = "Word frequencies based on tf-idf")
```




```
barplot(clus_terms[[ 3 ]][ order(clus_terms[[3]], decreasing = TRUE) ][ 1:20 ], las = 2, name
s.arg = names(clus_terms[[3]][ order(clus_terms[[3]], decreasing = TRUE) ][ 1:20]),col ='#a63
603', main ="Most frequent words in Cluster 3",
        ylab = "Word frequencies based on tf-idf")
```



Conclusion:

We saw that our corpus has a hierarchical structure from the dendograms and dividing the cluster groups into 3 gave us the most defined clusters of 3 different topics - computers, games and religion respectively. Thus, these 3 are the topics in the random 25% sample chosen from the corpus. The bar plots give us a good understanding on which words are most frequently related to the topics assigned. For instance, in computers windows is quite frequent which is expected since its the most widely used operating system. Similarly, in games we have hockey, baseball, fan as one of the most frequent terms whereas religion has mostly words such as sin, homosexual, gay, jesus etc which are common words used by many religious people.