

Text Mining

Assignment Number 1



April 9, 2018

Pooja Sheth

UW MSBA TACOMA

**Corpus1 - Health, Exercise, Fitness Data**

#Create Corpus

docs <- Corpus(DirSource("D:/Lenovo backup/UW/Q3/Text mining/health"))

docs

#inspect a particular document

writeLines(as.character(docs[[30]]))

getTransformations()

[1] "removeNumbers" "removePunctuation" "removeWords" "stemDocument" "stripWhitespace"

#preliminary clean-up steps beforetransformations

#inspect some documents in the corpus

#Find colons and hyphens without spaces between the words separated by them.

#Using the removePunctuation transform without fixing this will cause the two words on either side of the symbols to be combined.

#Fix this prior to using the transformations.

#create a custom transformation with tm package

#via the content\_transformer function

#which takes a function as input which specifies what transformation needs to be done.

#input function would be one that replaces all instances of a character by spaces.

#Use the gsub() function does just that.

#create the toSpace content transformer

toSpace <- content\_transformer(function(x, pattern) {return (gsub(pattern, " ", x))})

docs <- tm\_map(docs, toSpace, "-")

docs <- tm\_map(docs, toSpace, ":")

docs <- tm\_map(docs,toSpace, "–")

#Remove punctuation – replace punctuation marks with " “

writeLines(as.character(docs[[30]]))

docs <- tm\_map(docs, removePunctuation)

writeLines(as.character(docs[[30]]))

#inspect

docs <- tm\_map(docs, toSpace, "’")

docs <- tm\_map(docs, toSpace, "‘")

docs <- tm\_map(docs, toSpace, " -")

#Convert the corpus to lower case

#Transform to lower case (need to wrap in content\_transformer)

docs <- tm\_map(docs,content\_transformer(tolower))

#Remove all numbers.

#Strip digits (std transformation, so no need for content\_transformer)

docs <- tm\_map(docs, removeNumbers)

#remove stopwords using the standard list in tm

docs <- tm\_map(docs, removeWords, stopwords("english"))

#remove all extraneous whitespaces using the stripWhitespace transformation:

#Strip whitespace (cosmetic?)

docs <- tm\_map(docs, stripWhitespace)

#stemming

#common Root Words

# Simple steming algorithms cut off the ends - mate - mating to mat

writeLines(as.character(docs[[30]]))

#Stem document

docs <- tm\_map(docs,stemDocument)

writeLines(as.character(docs[[30]]))

#use lemmatization instead to take into account parts of speech - grammatical context

#look for problems or different spellings or variations

docs <- tm\_map(docs, content\_transformer(gsub), pattern = "interv", replacement = "interval")

docs <- tm\_map(docs, content\_transformer(gsub), pattern = "train", replacement = "training")

docs <- tm\_map(docs, content\_transformer(gsub), pattern = "elimin", replacement = "eliminate")

docs <- tm\_map(docs, content\_transformer(gsub), pattern = "experi", replacement = "experiment")

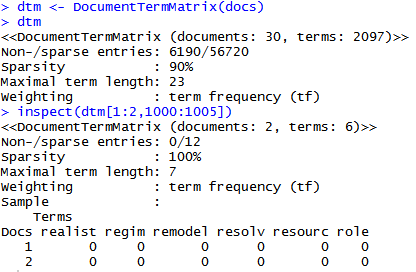
docs <- tm\_map(docs, content\_transformer(gsub), pattern = "intens", replacement = "intensity")

docs <- tm\_map(docs, content\_transformer(gsub), pattern = "exercis", replacement = "exercise ")

docs <- tm\_map(docs, content\_transformer(gsub), pattern = "fit", replacement = "fitness")

docs <- tm\_map(docs, content\_transformer(gsub), pattern = "activ", replacement = "active")

#document term matrix



#Mining the corpus

#first convert the TDM into a mathematical matrix using the as.matrix() function

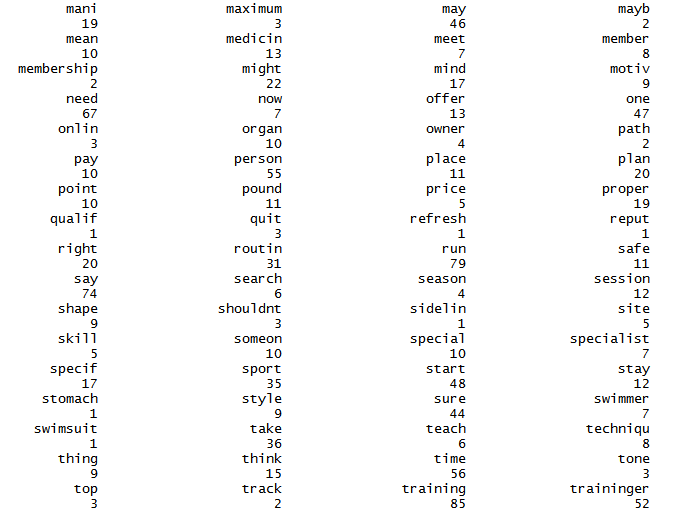
#then sum over all rows to give us the for each column (term)

#result is stored in the (column matrix) variable freq.

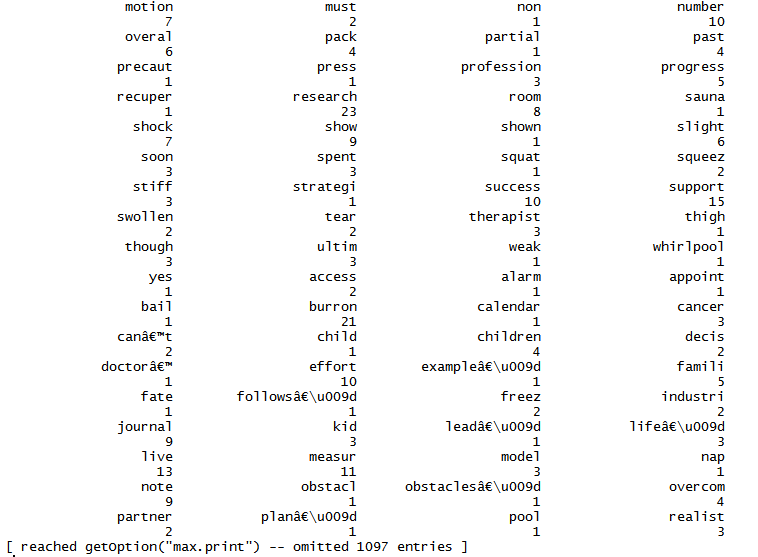
freq <- colSums(as.matrix(dtm))

freq









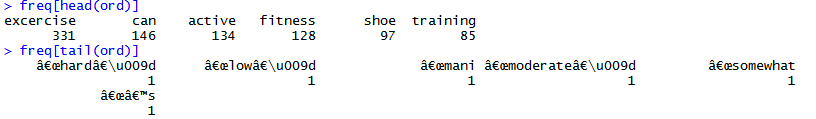
length(freq)

[1] 2097

#create sort order (descending)

ord <- order(freq,decreasing=TRUE)

#list the most and least frequently occurring terms:

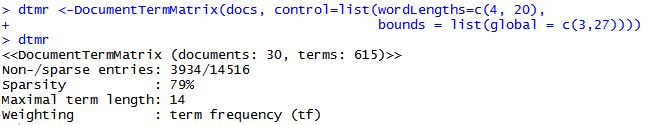


Couple of ways to simple ways to strike a balance between frequency and specificity.

#One way - use inverse document frequencies.

#A simpler approach - eliminate words that occur in a large fraction of corpus documents.

#keep only words that are longer than 3 characters and less than 21



> #review the frequencies as before

> freqr <- colSums(as.matrix(dtmr))

> #length should be total number of terms

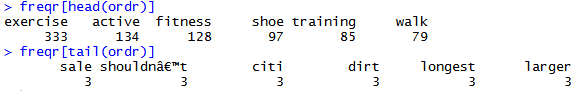
> length(freqr)

[1] 616

#create sort order (asc)

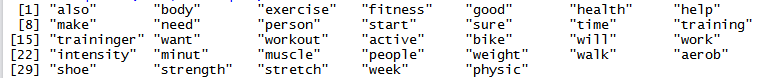
ordr <- order(freqr,decreasing=TRUE)

#inspect most frequently occurring terms



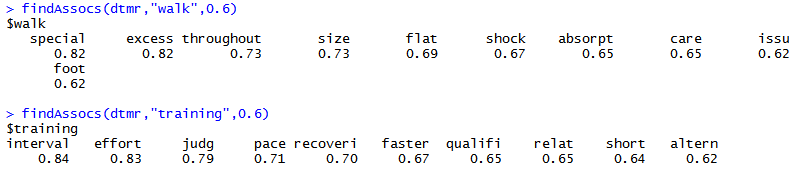
#find terms that have occured at least 70 times

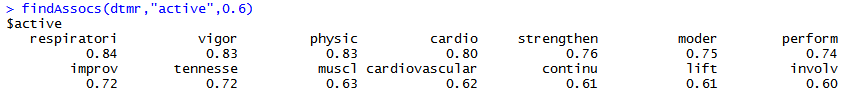
findFreqTerms(dtmr,lowfreq=70)



#find associations between frequent terms (in this case) - correlations between terms

#for findassocs provide the DTM, the term of interest and the correlation limit - between 0 and 1(always occur)





#presence of a term in these list is not indicative of its frequency

#it is a measure of the frequency with which the two (search and result term) co-occur (or show up together) in documents across

#Note also, that it is not an indicator of nearness or contiguity -

#because the document term matrix does not store any information on proximity of term - simply a “bag of words."

#lets look at frequency histogram

#create a data frame – a list of columns of equal length.

#the data frame also contains the name of the columns – in this case these are term and occurrence respectively

wf=data.frame(term=names(freqr),occurrences=freqr)

names(wf)

#plot only those terms that occur more than 100 times.

#the aes option describes plot aesthetics

#we use it to specify the x and y axis labels

#the stat="identity" option in geom\_bar () ensures that the height of each bar is

#proportional to the data value that is mapped to the y-axis (i.e occurrences)

#the last line specifies that the x-axis labels should be at a 45-degree angle and should be horizontally justified

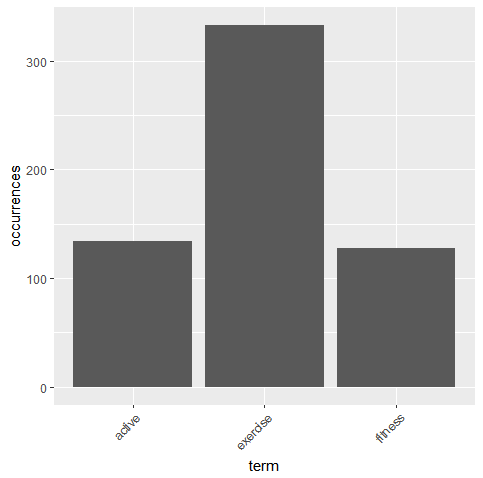
#(see what happens if you leave this out).

p <- ggplot(subset(wf, freqr>100), aes(term, occurrences))

p <- p + geom\_bar(stat="identity")

p <- p + theme(axis.text.x=element\_text(angle=45, hjust=1))

p



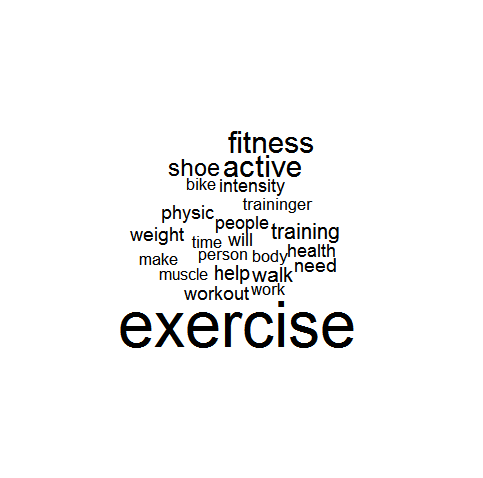
#Setting a seed number ensures that you get the same look each time

#The arguments of the wordcloud() function - can specify the maximum number of words to be included instead of the minimum frequency

set.seed(42)

#limit words by specifying min frequency

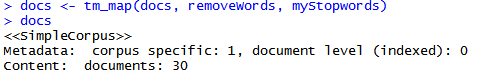
wordcloud(names(freqr),freqr, min.freq=50)



#…add color

wordcloud(names(freqr), freqr,min.freq=50,colors=brewer.pal(6,"Dark2"))





#clustering example

#convert dtm to matrix

m <- as.matrix(dtm)

#write as csv file (optional)

write.csv(m,file="dtmEight2Late.csv")

#shorten rownames for display purposes

rownames(m) <- paste(substring(rownames(m),1,3),rep("..",nrow(m)),substring(rownames(m), nchar(rownames(m))-12,nchar(rownames(m))-4))

#compute distance between document vectors

d <- dist(m)

#run hierarchical clustering using Ward’s method

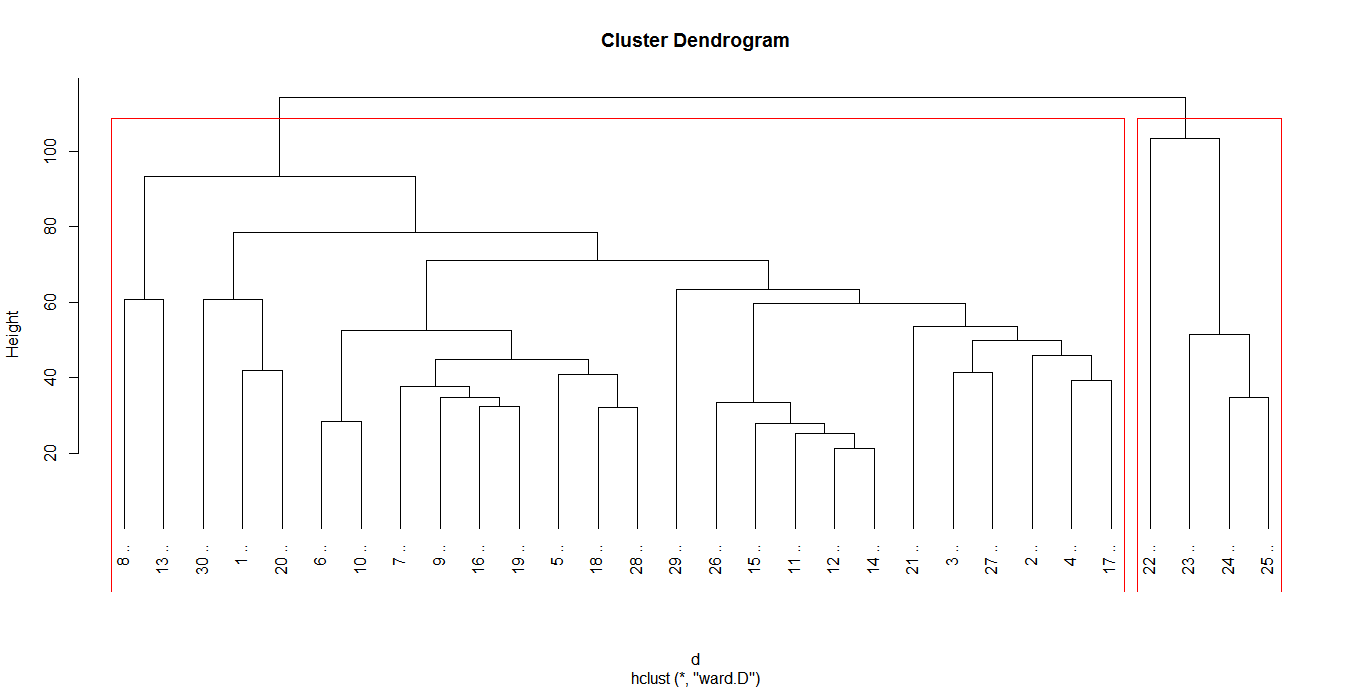
groups <- hclust(d,method="ward.D")

#plot dendogram, use hang to ensure that labels fall below tree

plot(groups, hang=-1)

#cut into 2 subtrees

rect.hclust(groups,2)

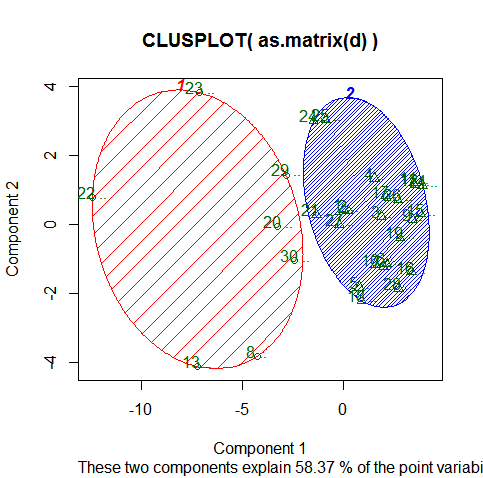


#k means algorithm, 2 clusters, 100 starting configurations

kfit <- kmeans(d, 2, nstart=100)

clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels=2, lines=0)

#kmeans – determine the optimum number of clusters (elbow method)

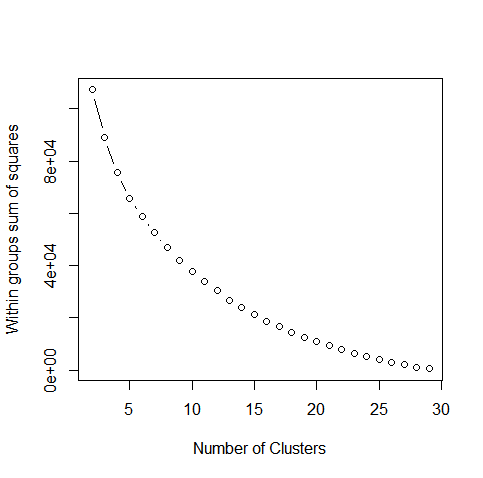


#look for “elbow” in plot of summed intra-cluster distances (withinss) as fn of k

wss <- 2:29

for (i in 2:29) wss[i] <- sum(kmeans(d,centers=i,nstart=25)$withinss)

plot(2:29, wss[2:29], type="b", xlab="Number of Clusters",ylab="Within groups sum of squares")



**Corpus2 - Organic Food Data**

#Create Corpus

docs <- Corpus(DirSource("D:/Lenovo backup/UW/Q3/Text mining/Food"))

docs

#inspect a particular document

writeLines(as.character(docs[[29]]))

getTransformations()

#preliminary clean-up steps beforetransformations

#inspect some documents in the corpus

#Find colons and hyphens without spaces between the words separated by them.

#Using the removePunctuation transform without fixing this will cause the two words on either side of the symbols to be combined.

#Fix this prior to using the transformations.

#create a custom transformation with tm package

#via the content\_transformer function

#which takes a function as input which specifies what transformation needs to be done.

#input function would be one that replaces all instances of a character by spaces.

#Use the gsub() function does just that.

#create the toSpace content transformer

toSpace <- content\_transformer(function(x, pattern) {return (gsub(pattern, " ", x))})

docs <- tm\_map(docs, toSpace, "-")

docs <- tm\_map(docs, toSpace, ":")

docs <- tm\_map(docs,toSpace, "–")

#Remove punctuation – replace punctuation marks with " “

writeLines(as.character(docs[[29]]))

docs <- tm\_map(docs, removePunctuation)

writeLines(as.character(docs[[29]]))

#inspect

docs <- tm\_map(docs, toSpace, "’")

docs <- tm\_map(docs, toSpace, "‘")

docs <- tm\_map(docs, toSpace, " -")

#Convert the corpus to lower case

#Transform to lower case (need to wrap in content\_transformer)

docs <- tm\_map(docs,content\_transformer(tolower))

#Remove all numbers.

#Strip digits (std transformation, so no need for content\_transformer)

docs <- tm\_map(docs, removeNumbers)

#remove stopwords using the standard list in tm

docs <- tm\_map(docs, removeWords, stopwords("english"))

#remove all extraneous whitespaces using the stripWhitespace transformation:

#Strip whitespace (cosmetic?)

docs <- tm\_map(docs, stripWhitespace)

#stemming

#common Root Words

# Simple steming algorithms cut off the ends - mate - mating to mat

writeLines(as.character(docs[[29]]))

#Stem document

docs <- tm\_map(docs,stemDocument)

writeLines(as.character(docs[[29]]))

#use lemmatization instead to take into account parts of speech - grammatical context

#look for problems or different spellings or variations

docs <- tm\_map(docs, content\_transformer(gsub), pattern = "agronomi", replacement = "agronomy")

docs <- tm\_map(docs, content\_transformer(gsub), pattern = "disord", replacement = "disorder")

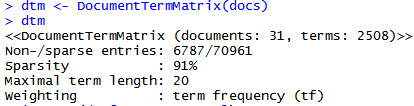
docs <- tm\_map(docs, content\_transformer(gsub), pattern = "chemic", replacement = "chemical")

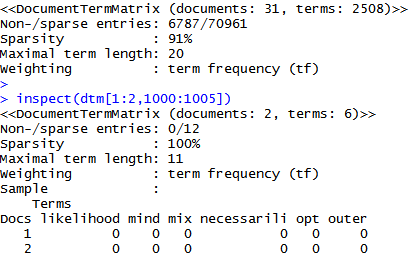
docs <- tm\_map(docs, content\_transformer(gsub), pattern = "damag", replacement = "damage")

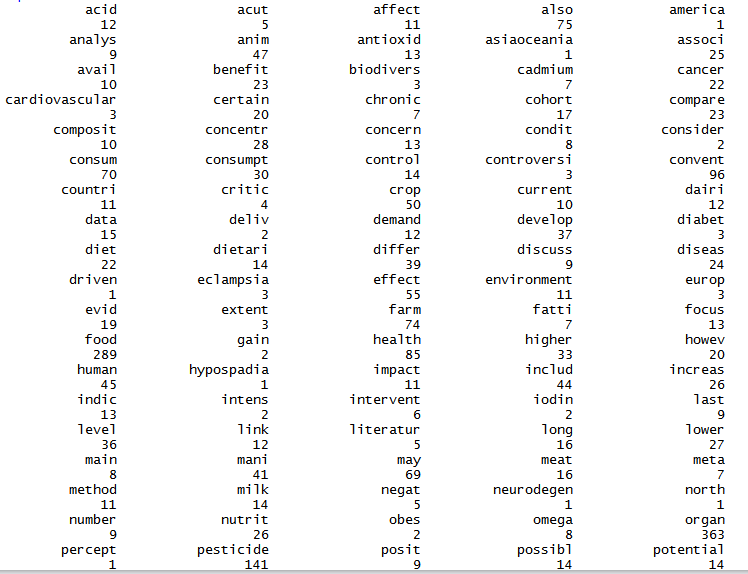
docs <- tm\_map(docs, content\_transformer(gsub), pattern = "pesticid", replacement = "pesticide")

docs <- tm\_map(docs, content\_transformer(gsub), pattern = "studi", replacement = "studio")

#document term matrix









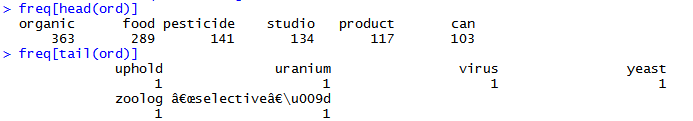
> length(freq)

[1] 2508

#create sort order (descending)

ord <- order(freq,decreasing=TRUE)

#list the most and least frequently occurring terms:

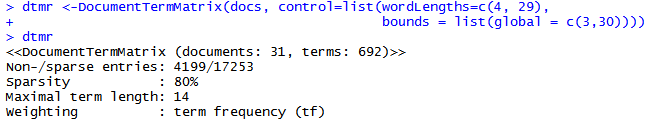


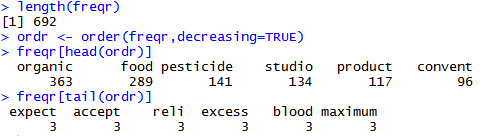
#Couple of ways to simple ways to strike a balance between frequency and specificity.

#One way - use inverse document frequencies.

#A simpler approach - eliminate words that occur in a large fraction of corpus documents.

#keep only words that are longer than 3 characters and less than 21

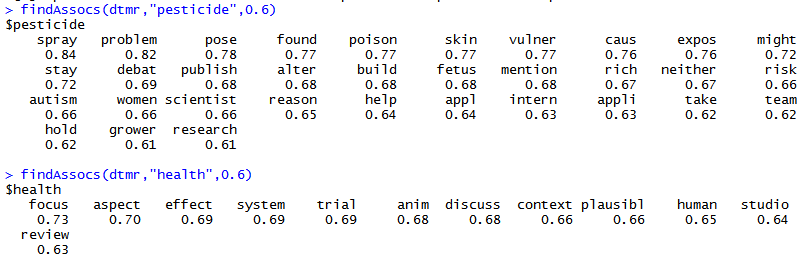


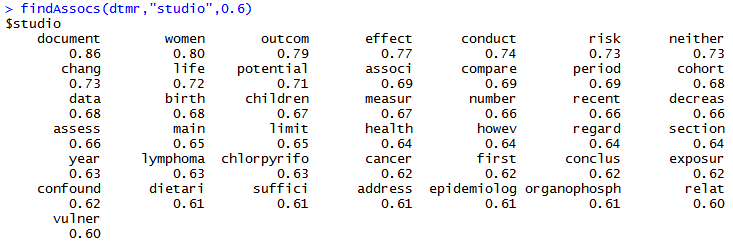




#find associations between frequent terms (in this case) - correlations between terms

#for findassocs provide the DTM, the term of interest and the correlation limit - between 0 and 1(always occur)





#presence of a term in these list is not indicative of its frequency

#it is a measure of the frequency with which the two (search and result term) co-occur (or show up together) in documents across

#Note also, that it is not an indicator of nearness or contiguity -

#because the document term matrix does not store any information on proximity of term - simply a “bag of words."

#let’s look at frequency histogram

#create a data frame – a list of columns of equal length.

#the data frame also contains the name of the columns – in this case these are term and occurrence respectively

wf=data.frame(term=names(freqr),occurrences=freqr)

names(wf)

[1] "term" "occurrences"

#plot only those terms that occur more than 100 times.

#the aes option describes plot aesthetics

#we use it to specify the x and y axis labels

#the stat="identity" option in geom\_bar () ensures that the height of each bar is

#proportional to the data value that is mapped to the y-axis (i.e occurrences)

#the last line specifies that the x-axis labels should be at a 45 degree angle and should be horizontally justified

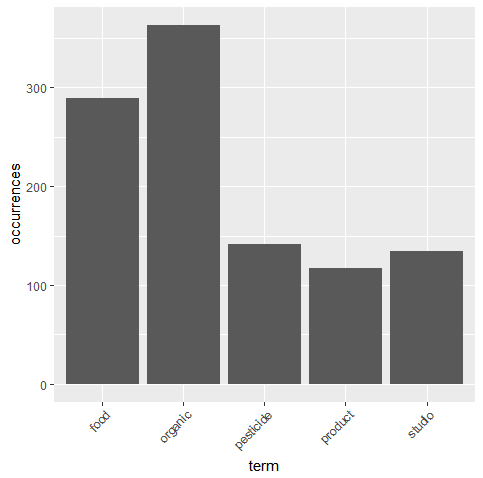
#(see what happens if you leave this out).

p <- ggplot(subset(wf, freqr>100), aes(term, occurrences))

p <- p + geom\_bar(stat="identity")

p <- p + theme(axis.text.x=element\_text(angle=45, hjust=1))

p



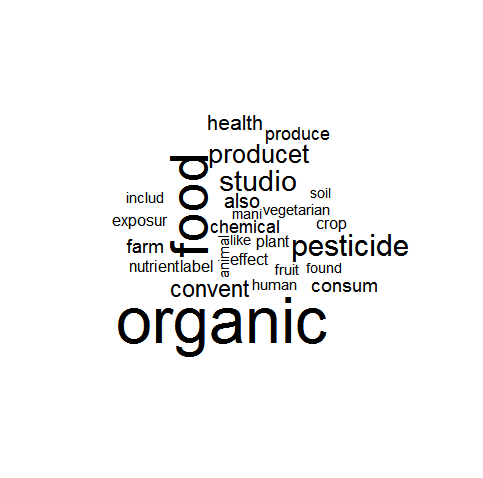
#Setting a seed number ensures that you get the same look each time

#The arguments of the wordcloud() function - can specify the maximum number of words to be included instead of the minimum frequency

set.seed(42)

#limit words by specifying min frequency

wordcloud(names(freqr),freqr, min.freq=70)



#…add color

wordcloud(names(freqr),freqr,min.freq=70,colors=brewer.pal(6,"Dark2"))



myStopwords <- c("can", "say","one","way","use",

"also","howev", "tell", "will", "found", "includ",

"much","need","take","tend","even",

"like","particular","rather","said",

"get","well","make","ask","come","end"<

"first","two","help","often","may",

"might","see","someth","thing","point",

"post", "look", "right", "now", "think", "'ve",

"'re ")

myStopwords <- c("can", "say","one","way","use",

"also","howev", "tell", "will", "found", "includ",

"much","need","take","tend","even",

"like","particular","rather","said",

"get","well","make","ask","come","end"<

"first","two","help","often","may",

"might","see","someth","thing","point",

"post", "look", "right", "now", "think", "'ve",

"'re ")

#remove custom stopwords

docs <- tm\_map(docs, removeWords, myStopwords)

docs

#clustering example

#convert dtm to matrix

m <- as.matrix(dtm)

#write as csv file (optional)

write.csv(m,file="dtmEight2Late.csv")

#shorten rownames for display purposes

rownames(m) <- paste(substring(rownames(m),1,3),rep("..",nrow(m)),substring(rownames(m), nchar(rownames(m))-12,nchar(rownames(m))-4))

#compute distance between document vectors

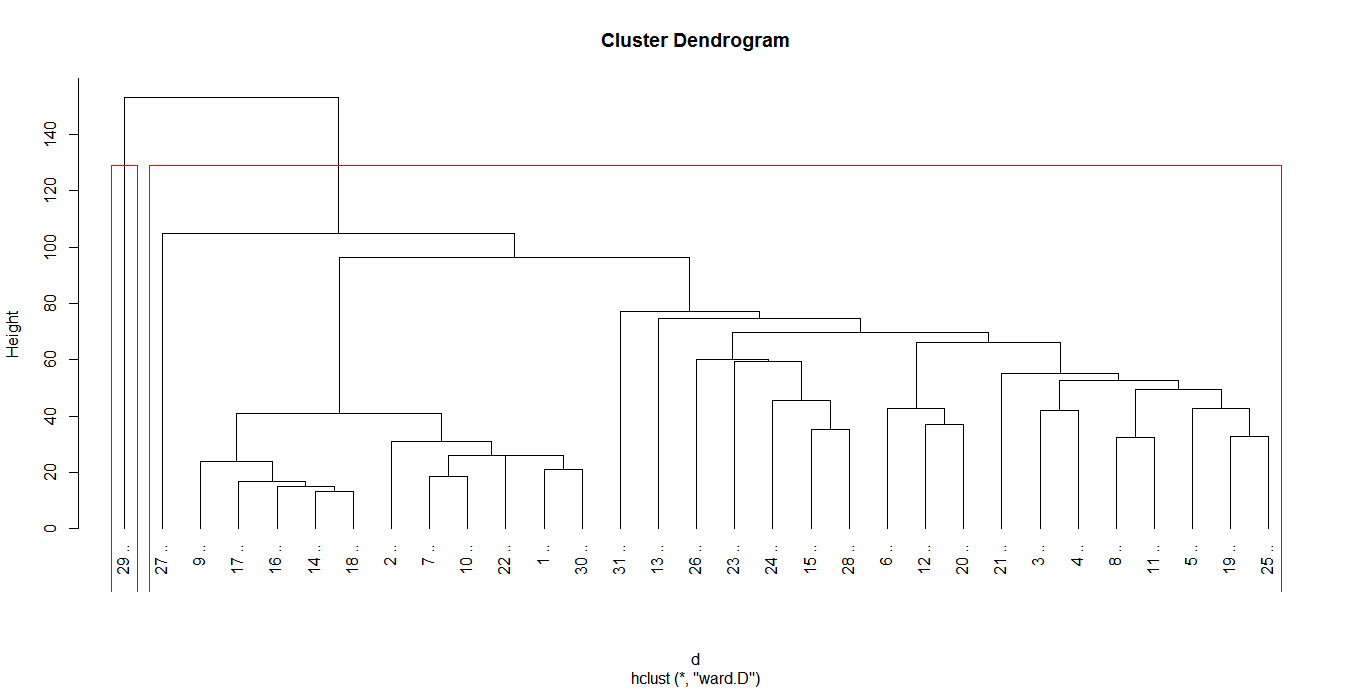
d <- dist(m)

#run hierarchical clustering using Ward’s method

groups <- hclust(d,method="ward.D")

#plot dendogram, use hang to ensure that labels fall below tree

plot(groups, hang=-1)

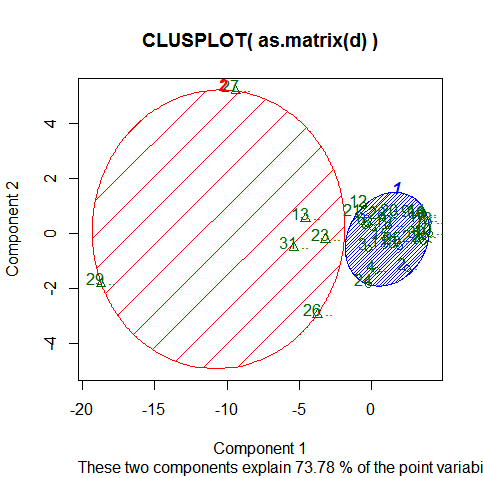


#k means algorithm, 2 clusters, 100 starting configurations

kfit <- kmeans(d, 2, nstart=100)

clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels=2, lines=0)

#kmeans – determine the optimum number of clusters (elbow method)



#look for “elbow” in plot of summed intra-cluster distances (withinss) as fn of k

wss <- 2:29

for (i in 2:29) wss[i] <- sum(kmeans(d,centers=i,nstart=25)$withinss)

plot(2:29, wss[2:29], type="b", xlab="Number of Clusters",ylab="Within groups sum of squares")

