Text Mining: Part 1

IBM Knowledge Center

Text mining is the **process of analyzing collections of textual materials** in order **to capture key concepts** and **themes** and **uncover hidden relationships** and **trends** without requiring that you know the precise words or terms that authors used to express those concepts.

1. Web Scraping and Tidy Text
2. Text Mining with Multiple Text Datasets

1. Web Scraping and Tidy Text

https://www.tidytextmining.com

Tidy text format is **a table with one-token-per-row**. A **token** is a meaningful unit of text, which is often a **single word**, but can also be an **n-gram** (a list of n words), sentence, or **paragraph**. **Tokenization** refers to the process of splitting text into tokens.

[Data] President Trump's Remarks to the South Korean National Assembly

Full Text of President Trump's Remarks to the South Korean National Assembly

November 07, 2017

Remarks by President Trump to the National Assembly of the Republic of Korea | Seoul, Republic of Korea

National Assembly Building

Seoul, Republic of Korea

11:24 A.M. KST

PRESIDENT TRUMP: Assembly Speaker Chung, distinguished members of this Assembly, ladies and gentlemen: Thank you for the extraordinary privilege to speak in this great chamber and to address your people on behalf of the people of the United States of America.

In our short time in your country, Melania and I have been awed by its ancient and modern wonders, and we your welcome.

Together, we dream of a Korea that is free, a peninsula that is safe, and families that are reunited once again. We dream of highways connecting North and South, of cousins embracing cousins, and this nuclear nightmare replaced with the beautiful promise of peace.

Until that day comes, we stand strong and alert. Our eyes are fixed to the North, and our hearts praying for the day when all Koreans can live in freedom. (Applause.)

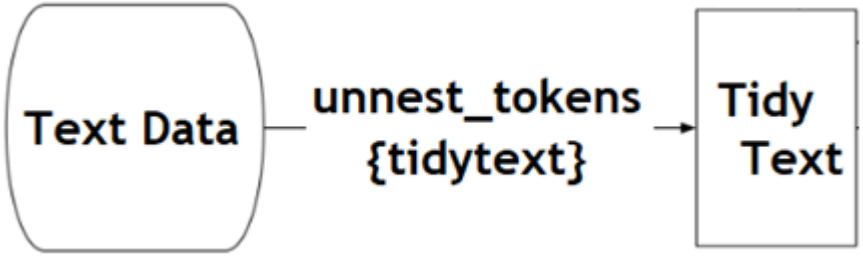
Thank you. (Applause.) God Bless You. God Bless the Korean people. Thank you very much. Thank you. (Applause.)

END

11:59 A.M. KST

[Step 1] Download text data from the web page

```
web <- paste0("https://www.voanews.com/a/text-of-trump-",
 "speech-to-south-korean-national-assembly-/4106294.html")
```




```
> # Method to get tokens ###
> library(dplyr); library(readr); library(tidytext)
> tokens <- tibble(text=read_lines(web)) %>%
+   unnest_tokens(word, text, format="html")
> head(tokens,5)
# A tibble: 5 x 1
  word
  <chr>
1 full
2 text
3 of
4 president
5 trump
```

tibble: create data frame
unnest_tokens: tokenize one-token-per-row

In case you get an error handling 'word', find out which packages are loaded by typing "*search()*" and unload *Rmisc* or *plyr*. If this fails, try re-installing the package: *install.packages("tidytext")*.

[Step 2] Cleaning and sorting of text data

Stop words are extremely common words such as “the”, “of”, and “to”.

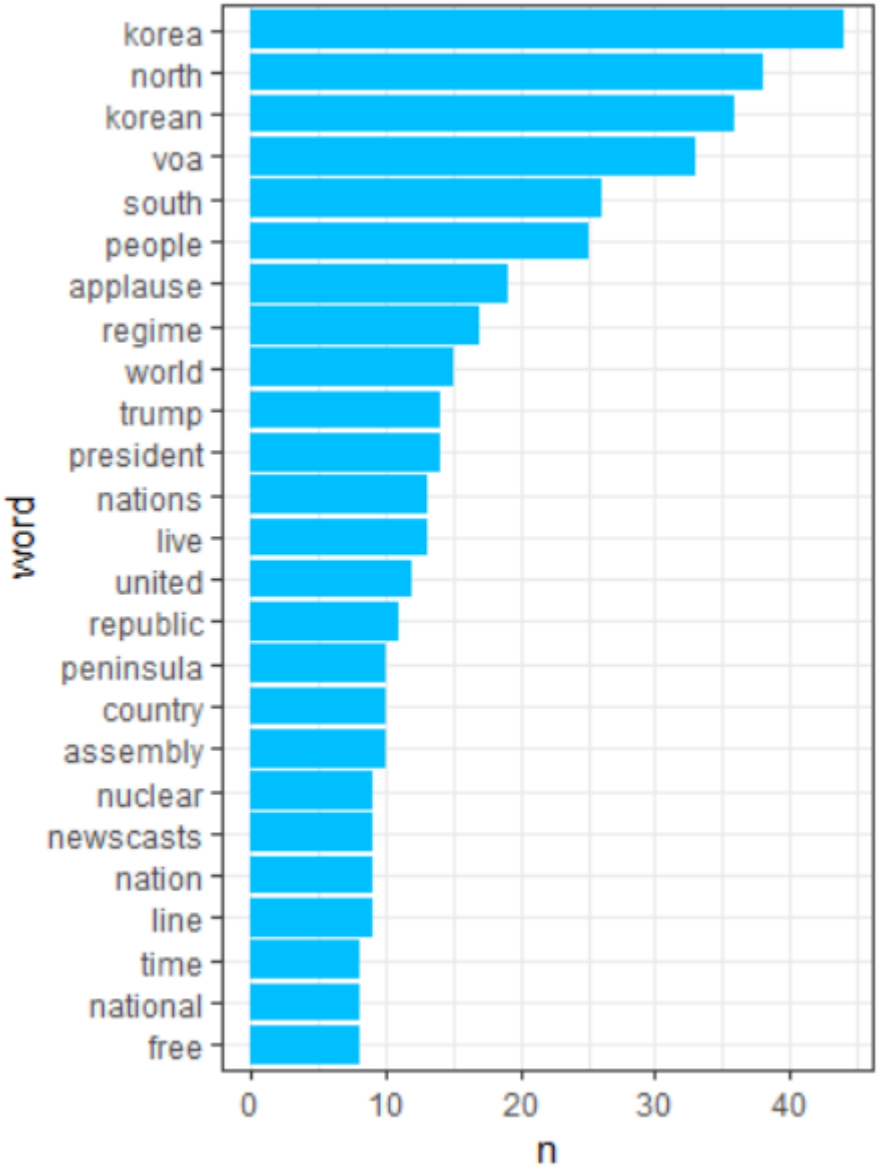
count(x, ..., sort = FALSE) {dplyr}
Count observations by group

```
> wcorp <- tokens %>%
+   anti_join(stop_words) %>%
+   count(word, sort=TRUE)
Joining, by = "word"
> wcorp
# A tibble: 1,061 x 2
   word          n
   <chr>     <int>
 1 korea        45
 2 north        38
 3 korean       36
 4 voa          36
 5 people       27
 6 south        25
 7 applause     19
 8 regime       17
 9 world        15
10 live         13
# ... with 1,051 more rows
```

```
#Removing some words
rwords <- tibble(word=c("day","line","live","voa",
                        "applause","trump","english","newscasts"))
wcorp <- wcorp %>% anti_join(rwords)
```

[Step 3] Plot word frequency


```
library(ggplot2)
wcorp[1:25,] %>%
  mutate(word=reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_bar(stat="identity",fill="deepskyblue") +
  theme_bw() +
  coord_flip()
```

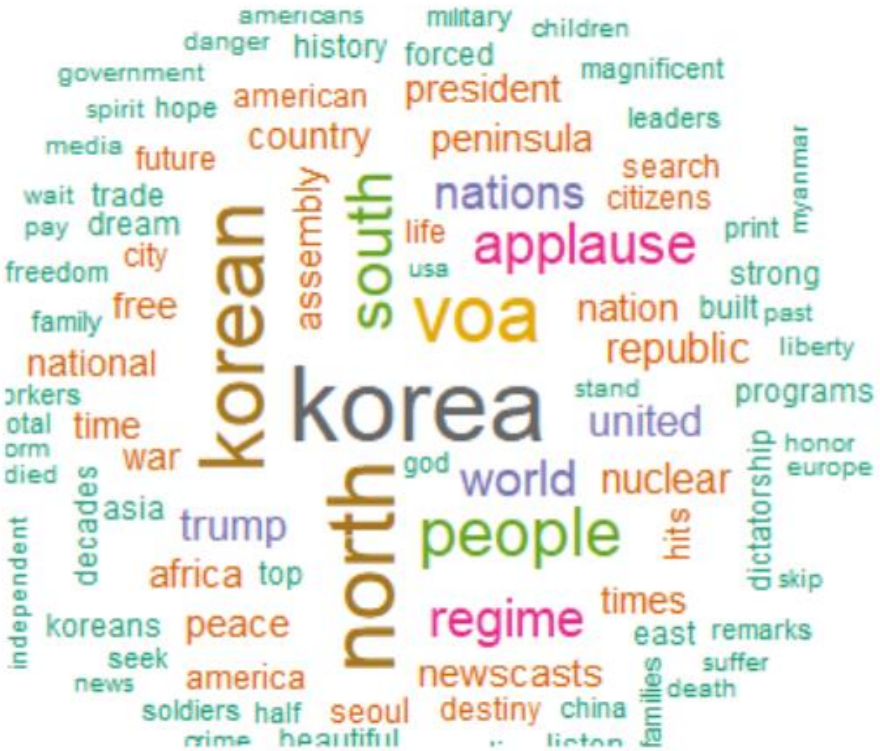


[Step 4] Generate the Word Cloud

The importance of words can be illustrated as a **word cloud.**

wordcloud(words,freq,min.freq=3,random.order,colors, ...) {wordcloud}
Plot a word cloud

```
library(wordcloud)
library(RColorBrewer)
wcorp %>%
  with(wordcloud(word,n,scale=c(3,0.5),max.words=100,
       colors=brewer.pal(8,"Dark2"),random.order=FALSE))
```



- **Qualitative palettes** employ different hues to create visual differences between classes. These palettes are suggested for nominal or categorical data sets.

Palette info at https://tinyurl.com/y3bsdffs

2. Text Mining with Multiple Text Datasets

The Fourth Industrial Revolution: what it means, how to respond

14 Jan 2016

Klaus Schwab

Schwab.txt

32,883 views | Aug 13, 2018, 12:26am

The 4th Industrial Revolution Is Here - Are You Ready?

Bernard Marr Contributor ⓘ

Enterprise & Cloud

Marr.txt

The Fourth Industrial Revolution and the factories of the future

By Enno de Boer

Boer.txt

[Step 1] Loading Text Data into R

```
> df <- file.path("./Data")
> dir(df)
[1] "Boer.txt"   "Marr.txt"   "Schwab.txt"
```

```
> library(NLP); library(tm)
> docs <- VCorpus(DirSource(df))
> summary(docs)
           Length Class             Mode
Boer.txt   2      PlainTextDocument list
Marr.txt   2      PlainTextDocument list
Schwab.txt 2      PlainTextDocument list
```

DirSource: create a directory source
(may need *encoding="UTF-8"* option)
VCorpus: create volatile corpora
(R objects held fully in memory)
PCorpus: permanent corpus

```
> inspect(docs)
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 3

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 12038

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 6683

[[3]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 16929

> inspect(docs[1])
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 1

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 12038

> inspect(docs[[1]])
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 12038

The Fourth Industrial Revolution and the factories of the future
Only a minority of manufacturers are achieving scale with the te
```

[Step 2] Cleaning and Counting Text Data

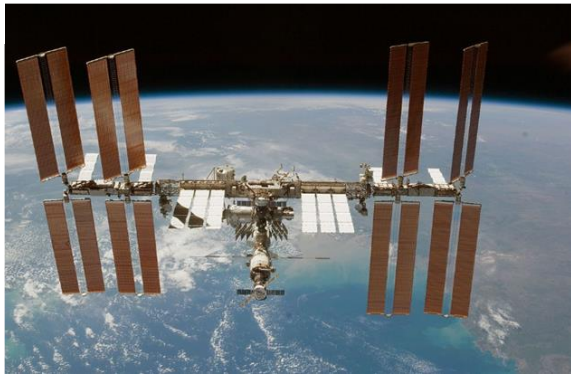
tidy(x, ...) {tidytext}
Tidy a Corpus object from the tm package

tidy: Turn an object into a tidy tibble

```
> # Split a column into tokens
> library(dplyr); library(tidytext)
> dxt <- tidy(docs) %>% unnest_tokens(word,text)
> dxt %>% top_n(4)
Selecting by word
# A tibble: 4 x 8
  author datetimestamp        description heading id    language origin word
  <lgl>  <dttm>               <lgl>       <lgl>   <chr> <chr>    <lgl>  <ch>
1 NA     2018-12-08 09:49:14 NA          NA      Boer~ en       NA     yie~
2 NA     2018-12-08 09:49:14 NA          NA      Boer~ en       NA     yie~
3 NA     2018-12-08 09:49:14 NA          NA      Marr~ en       NA     you
4 NA     2018-12-08 09:49:14 NA          NA      Schw~ en       NA     yie~
```

gsub(pattern, replacement, x, ...) {base}
This function replaces all matches of a string, if the parameter is a string vector, returns a string vector of the same length and with the same attributes.

```
#Replace "technologies" -> "technology"
dxt$word <- gsub("technologies","technology",dxt$word)
#Replace "sites" -> "site"
dxt$word <- gsub("sites","site",dxt$word)
#Remove numbers
dxt$word <- gsub('[[:digit:]]+', '', dxt$word)
```



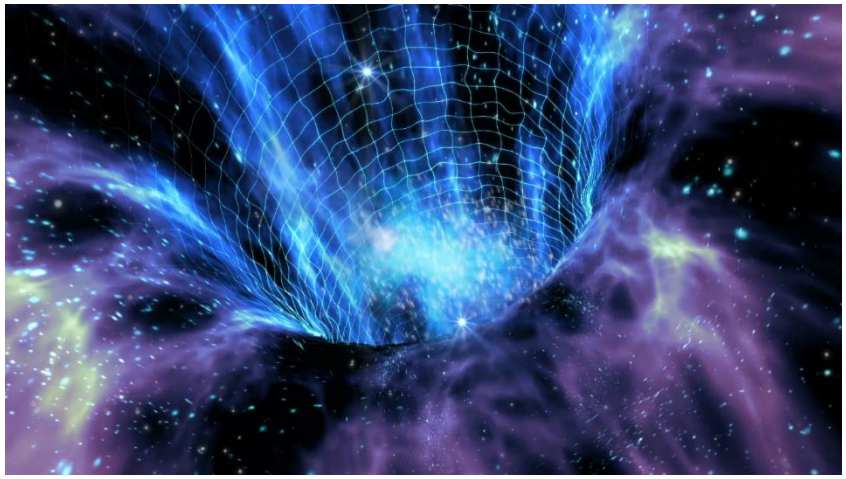
In text analysis, we need to remove stop words; stop words are words that are not useful for an analysis, typically extremely common words such as “the”, “of”, “to”, and so forth in English.

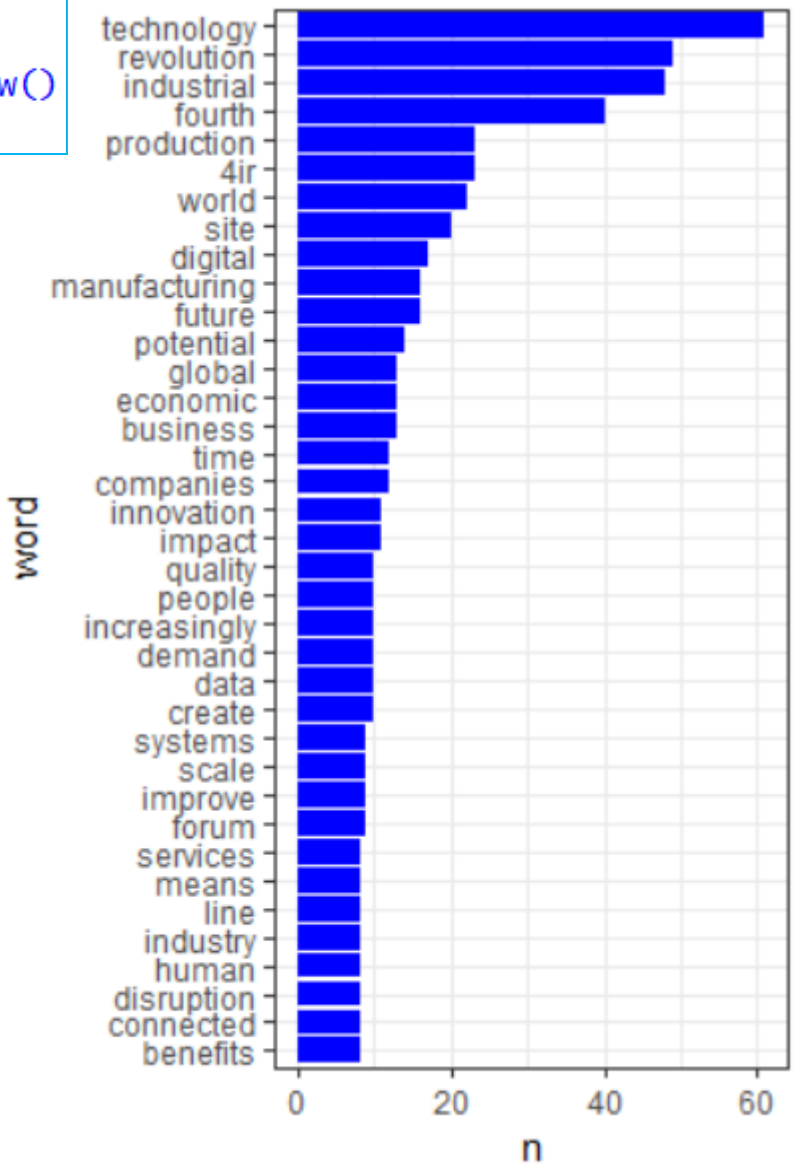
```
> #data(word,n)
> dxta <- dxt %>%
+   anti_join(stop_words) %>%
+   count(word, sort=TRUE)
Joining, by = "word"
> dxta
# A tibble: 1,276 x 2
   word            n
   <chr>       <int>
 1 technology     61
 2 revolution     49
 3 industrial     48
 4 fourth         40
 5 4ir            23
 6 production     23
 7 world          22
 8 site           20
 9 1              19
10 digital        17
```

```
> #data(id,word,n)
> dxtb <- dxt %>%
+   anti_join(stop_words) %>%
+   group_by(id) %>% count(word, sort=TRUE)
Joining, by = "word"
> dxtb %>% head(4)
# A tibble: 4 x 3
# Groups:   id [2]
  id        word            n
  <chr>     <chr>       <int>
1 Boer.txt technology     25
2 Boer.txt 4ir            23
3 Marr.txt industrial     23
4 Marr.txt revolution     23
```

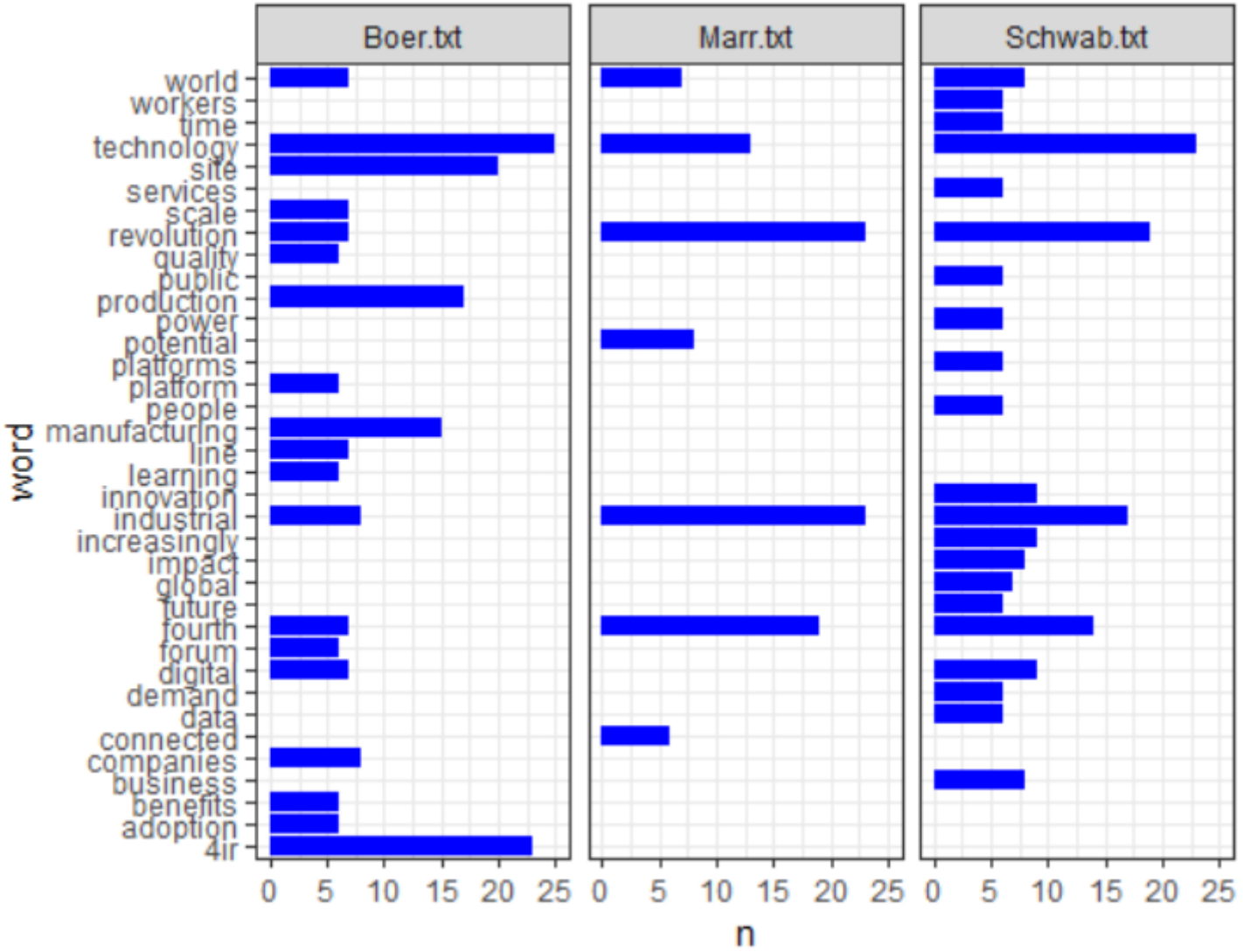
[Step 3] Visualizing Word Frequencies

```
> library(ggplot2)
> dxta %>% filter(n > 7) %>%
+   mutate(word=reorder(word, n)) %>%
+   ggplot(aes(word, n)) +
+   geom_bar(stat="identity",fill="blue") + theme_bw()
+   coord_flip()
```

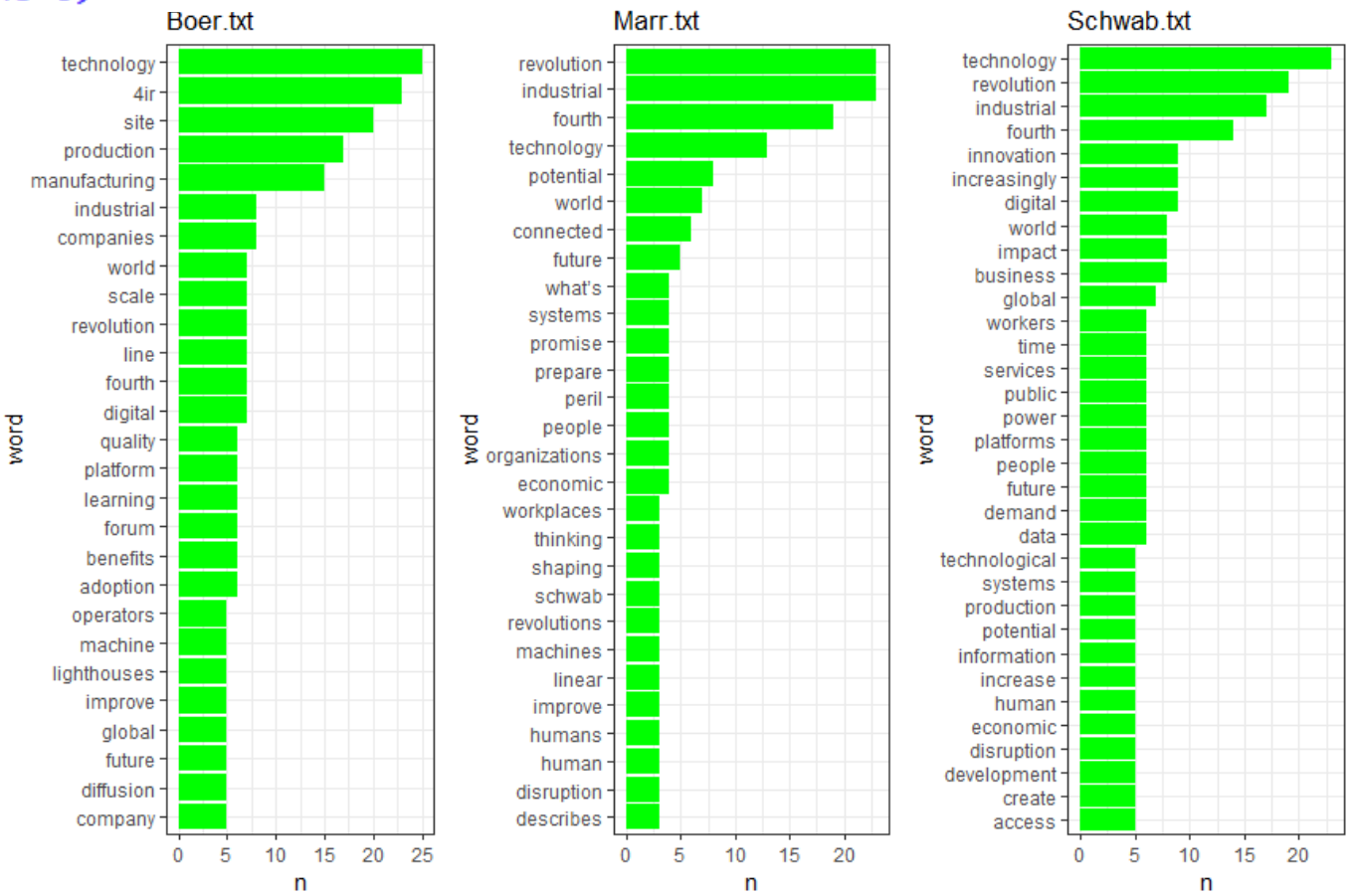




```
dxtb %>% filter(n > 5) %>%
  ggplot(aes(word, n)) +
  geom_bar(stat="identity",fill="blue") +
  facet_grid( ~ id) +  theme_bw() +
  coord_flip()
```



```
> gbar <- function(dat,nf,ids)
+ { dat %>% filter(n > nf, id==ids) %>%
+     mutate(word=reorder(word, n)) %>%
+     ggplot(aes(word, n)) + ggtitle(ids) +
+     geom_bar(stat="identity",fill="green") +
+     theme_bw() + coord_flip()
+ }
> g1 <- gbar(dxtb,4,"Boer.txt")
> g2 <- gbar(dxtb,2,"Marr.txt")
> g3 <- gbar(dxtb,4,"Schwab.txt")
> library(Rmisc)
> multiplot(g1,g2,g3,cols=3)
```



[Step 4] Word Clouds

wordcloud(words,freq,min.freq=3,max.words=Inf, ...) {wordcloud}

Plot a word cloud.

```
library(wordcloud)
dxta %>%
  with(wordcloud(word, n, max.words=60, colors=1:60))
```

max.words: Max # of words to be plotted

acast(data, formula, ...) {reshape2}

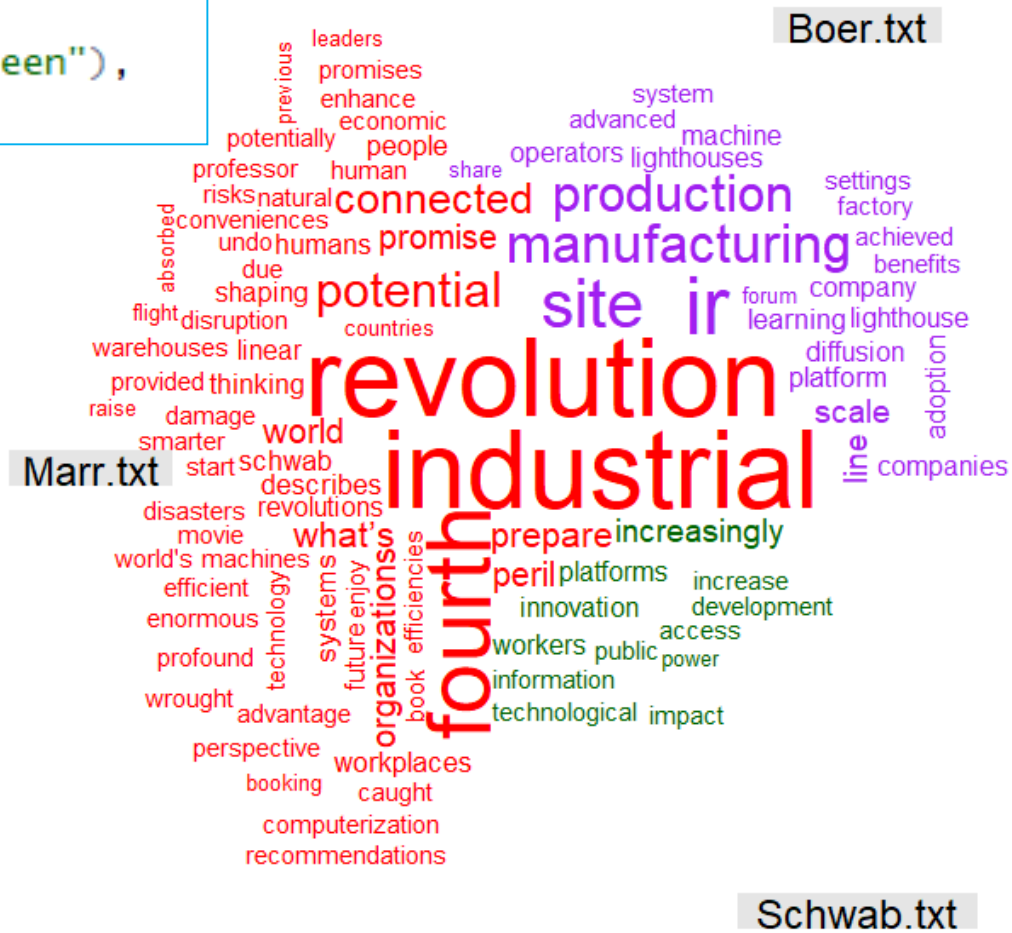
Cast a molten data frame into an array or data frame.

comparison.cloud(term.matrix, ...) {wordcloud}

Plot a cloud comparing the frequencies of words across documents.

```
library(reshape2)
dxtb %>%
  acast(word ~ id, value.var="n", fill=0) %>%
  comparison.cloud(random.order=FALSE,
      colors=c("purple", "red", "darkgreen"),
      title.size=1.5,max.words=100)
```

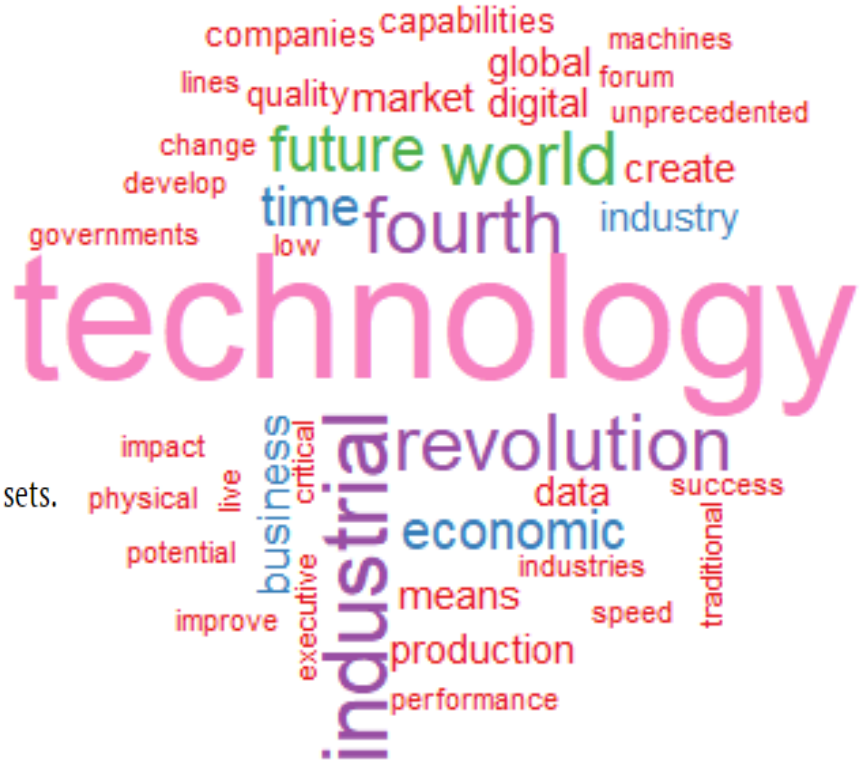
title.size: Input file name



commonality.cloud(term.matrix,comonality.measure=min,max.words=300,...) {wordcloud}

Plot a cloud of words shared across documents

```
library(reshape2)
set1 <- brewer.pal(8, "Set1")
dxtb %>%
  acast(word ~ id, value.var="n", fill=0) %>%
  commonality.cloud(colors=set1,max.words=50,
                    random.order=FALSE)
```



- **Qualitative palettes** employ different hues to create visual differences between classes. These palettes are suggested for nominal or categorical data sets.

[Step 5] Hierarchical Clustering of Words

Cleaning corpus functions:

stripWhitespace - removes extra whitespace from our corpus

removePunctuation - removes punctuation marks from our corpus

removeNumbers - remove numbers from our corpus

tolower - convert every word to a lower case

removeWords - removes stop words using stopwords("en") library

stemDocument - stems words in our corpus using Porter's stemming algorithm

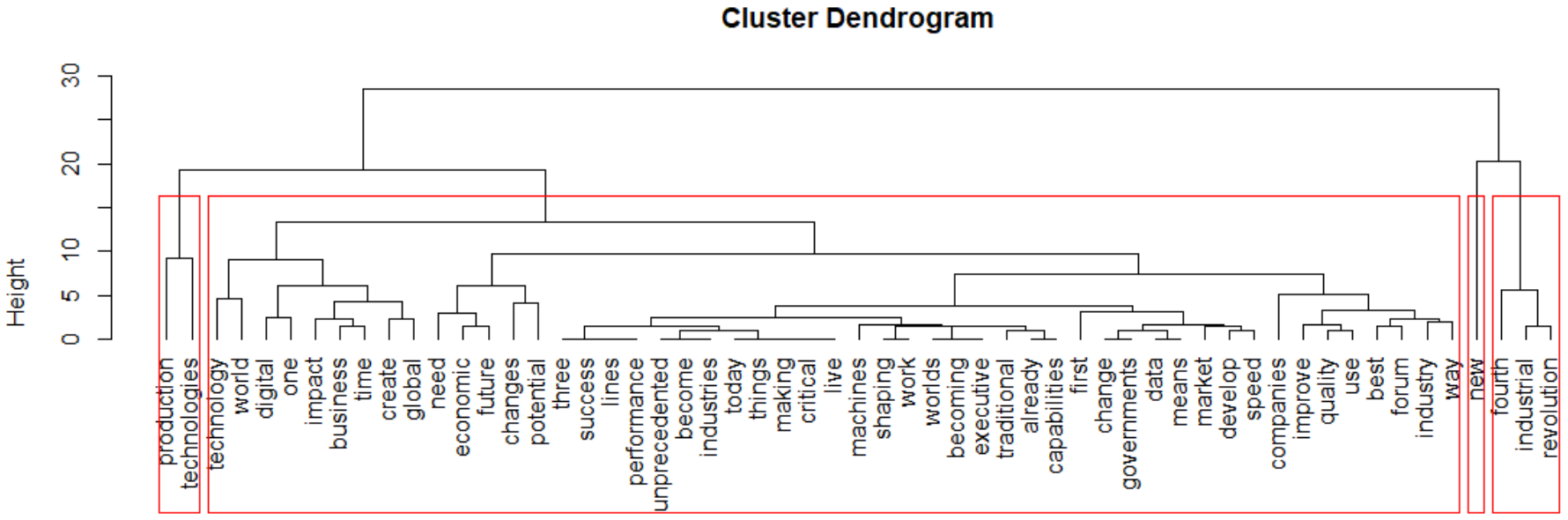
```
corp = tm_map(docs,removePunctuation)
corp = tm_map(corp,tolower)
corp = tm_map(corp,removeWords, stopwords("english"))
corp = tm_map(corp, removeWords,c("also","can","even","every",
                                  "made","well","will"))
corp = tm_map(corp,removeNumbers)
corp = tm_map(corp, stripWhitespace)
corp = tm_map(corp,PlainTextDocument)

removePUNCT <- content_transformer(
  function(x) gsub('[[:punct:][:blank:]]+'," ",x))
corp <- tm_map(corp, removePUNCT)
```

The **Document-Term Matrix (DTM)** describes the **frequency of terms** that occur in a collection of documents. To reduce the dimension of DTM, remove all popular items and instead utilize **"less frequent" terms**.

```
# Build a Term-Document Matrix
dtm <- DocumentTermMatrix(corp)
dtm = removeSparseTerms(dtm,0.15)
```

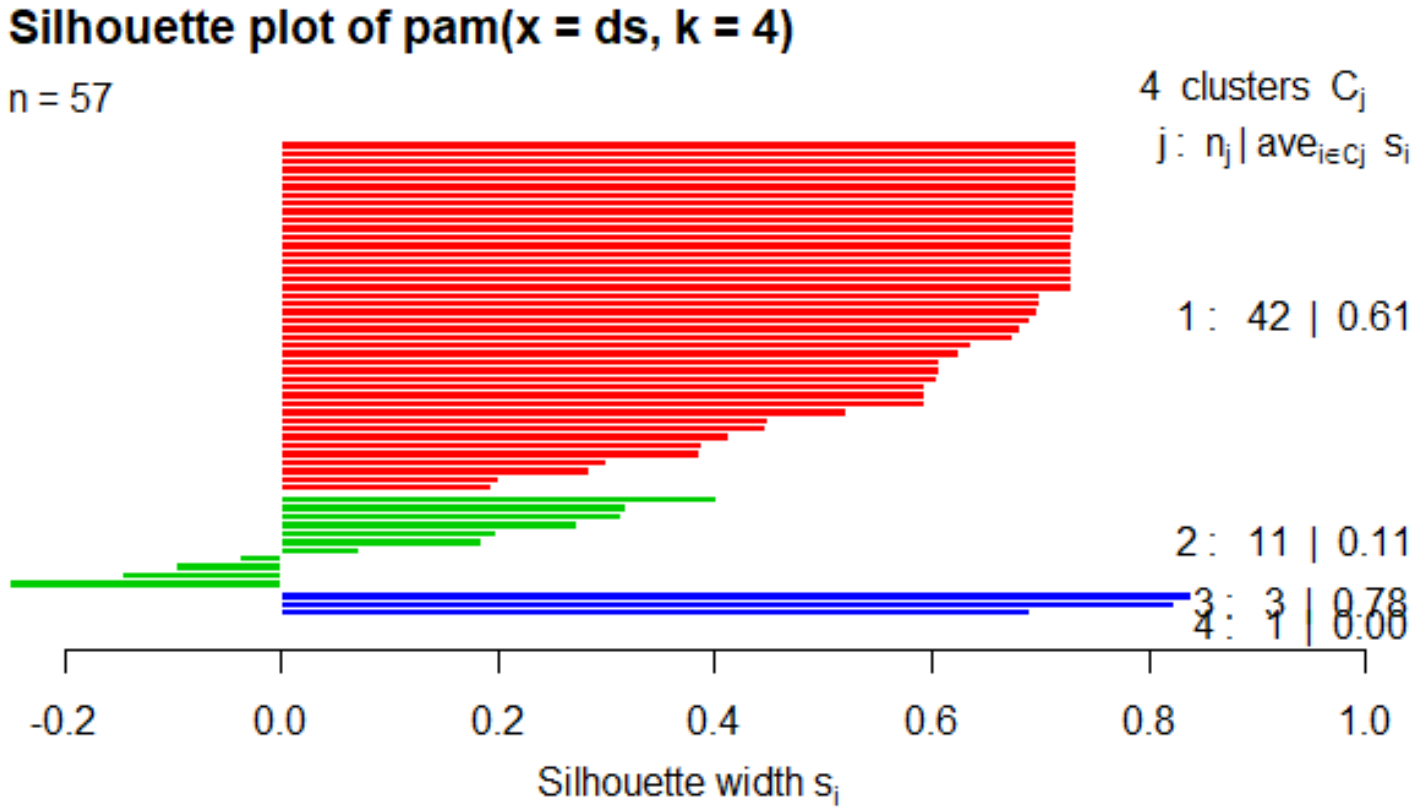
```
library(cluster)
ds = dist(t(dtm),method="euclidean")
fit = hclust(ds,method="complete")
plot(fit,hang=-1); rect.hclust(fit,k=4,border=2)
```

Hierarchical clustering based on Euclidean distance of terms



[Step 6] Silhouette Extract Information From Clustering


```
pam0 = pam(ds,4); si0 = silhouette(pam0); summary(si0)
plot(si0,col=2:5)
```



Average silhouette width : 0.51

[Step 7] K-Means Clustering

```
kfit = kmeans(ds,4)
table(kfit$cluster)
clusplot(as.matrix(ds), kfit$cluster, color=TRUE, shade=TRUE,
         labels=2, lines=0,cex=1)
```