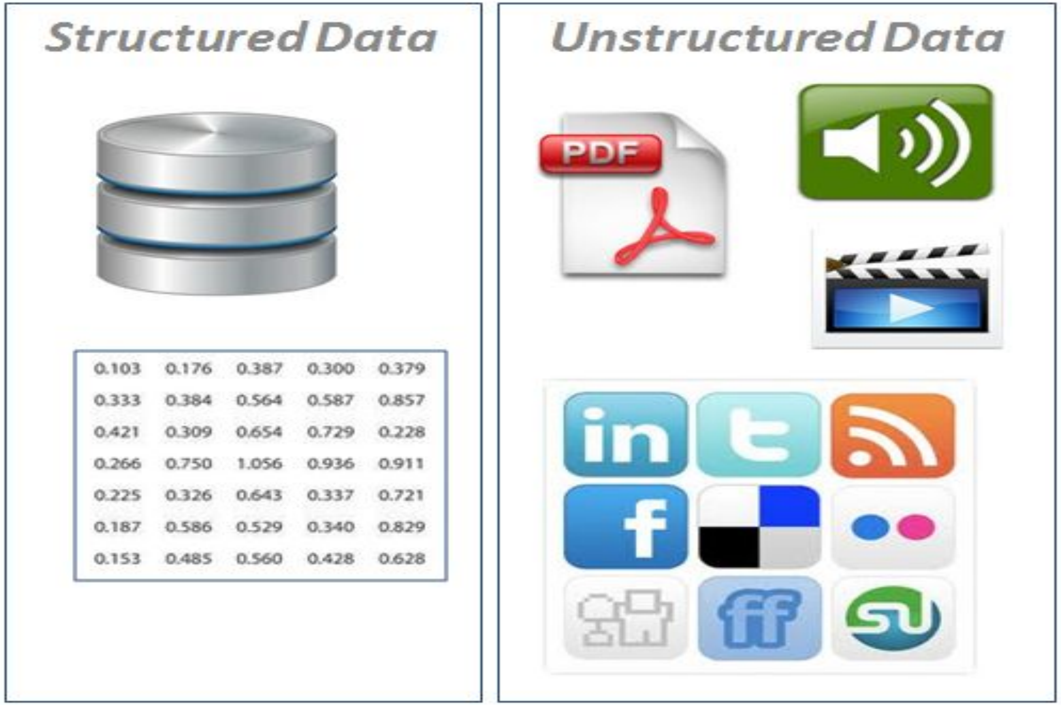
Text Mining - 1

Contents

1. Structured and Unstructured Data
2. Features of Unstructured Data
3. What is Text Analysis
4. Case Study
5. Text Mining in R
6. Word Cloud in R
7. Text Mining Using ggplot2

Structured Vs. Unstructured Data



0.103	0.176	0.387	0.300	0.379
0.333	0.384	0.564	0.587	0.857
0.421	0.309	0.654	0.729	0.228
0.266	0.750	1.056	0.936	0.911
0.225	0.326	0.643	0.337	0.721
0.187	0.586	0.529	0.340	0.829
0.153	0.485	0.560	0.428	0.628

Features Of Unstructured Data

Does not reside in traditional databases and data warehouses

May have an internal structure, but does not fit a relational data model

Generated by both humans and machines

- Textual and social media content
- Machine-to-machine communication

Examples Of Unstructured Data

Examples of unstructured data include:

- Personal messaging – Email, instant messages, tweets, chat
- Business documents – Business reports, presentations, survey responses
- Web content – Web pages, blogs, wikis, audio files, photos, videos
- Sensor output – Satellite imagery, geo-location data, scanner transactions

Value Of Unstructured Data

Unstructured data provides a rich source of information about people, households and economies

- It may enable more accurate and timely measurement of a range of demographic, social, economic and environmental phenomena
  - When combined with traditional data sources
  - As a replacement for traditional data sources

- As a result, presents unprecedented opportunities for official statistics to
  - Improve delivery of current statistical outputs
  - Create new information products not possible with traditional data sources

What Is Text Analysis

- Text Mining is also known as **Text Data Mining (TDM)** and **Knowledge Discovery in Textual Database (KDT)**

- It is a process of identifying novel information from a collection of texts (Also known as a ‘Corpus’)

- **Corpus is a collection of ‘documents’ containing natural language text.** Here, documents, generally, are sentences. Each document is represented as a separate line.

Case Study – HR Appraisal Process Feedback

Background

- The company XYZ carried out Annual Performance Appraisal process which is a routine HR process.
- The employees were asked to give feedback about the overall process and questions used for assessing their performance level.

Objective

- To understand the employee sentiments and incorporate recommendations in the current performance appraisal process.

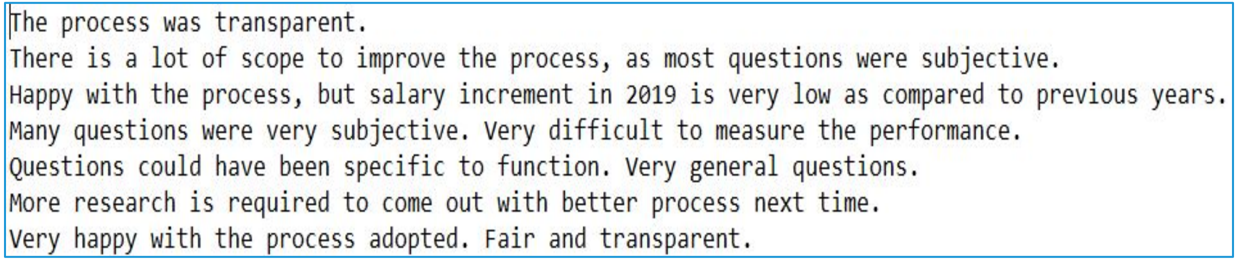
Available Information

- Feedback and comments from the employees were stored in a text document.

Data Snapshot

Example of data

Text Observations



The process was transparent.
There is a lot of scope to improve the process, as most questions were subjective.
Happy with the process, but salary increment in 2019 is very low as compared to previous years.
Many questions were very subjective. Very difficult to measure the performance.
Questions could have been specific to function. Very general questions.
More research is required to come out with better process next time.
Very happy with the process adopted. Fair and transparent.

* These are the comments received from employees.
Note that, data is not in structured format.

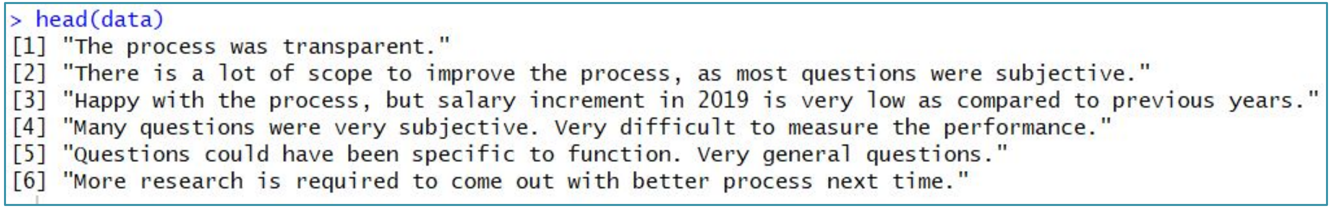
Text Mining In R

```
#Import the data.
#Import text file with one text record in one row
```

```
data<-readLines("HR Appraisal process.txt")
head(data)
```

❑ **readLines**() reads some or all text lines from a file or connection.

```
# Output:
```



```
> head(data)
[1] "The process was transparent."
[2] "There is a lot of scope to improve the process, as most questions were subjective."
[3] "Happy with the process, but salary increment in 2019 is very low as compared to previous years."
[4] "Many questions were very subjective. Very difficult to measure the performance."
[5] "Questions could have been specific to function. Very general questions."
[6] "More research is required to come out with better process next time."
```

Interpretation:

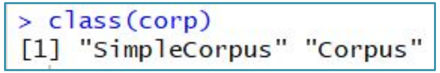
▯ **head()** prints first 6 text lines from the data with each line as one document / observation.

Text Mining In R

```
#Convert this data into ‘Corpus’
```

```
install.packages("tm")
library(tm)

corp <- Corpus(VectorSource(data))
class(corp)
```



```
> class(corp)
[1] "SimpleCorpus" "Corpus"
```

- ❑ Install and load **T**ext **M**ining (**tm**) package.
- ❑ **Vector source()** interprets each element of the vector as a document.
- ❑ **Corpus()** converts and saves data as a corpus.

Interpretation:

- Class of the data should be Corpus.

*

In case NLP is not loaded , Before installing tm , please run the following command

```
install.packages("NLP")
library(NLP)
```

Text Mining In R

```
# Inspect Corpus. Here [1:3] displays first 3 textlines.
```

```r
inspect(corp[1:3])
```

```
<<SimpleCorpus>>
Metadata: corpus specific: 1, document level (indexed): 0
Content: documents: 3

[1] The process was transparent.
[2] There is a lot of scope to improve the process, as most questions
were subjective.
[3] Happy with the process, but salary increment in 2019 is very low as
compared to previous years.
```

```
# Display a particular document from corpus.
```

```r
writeLines(as.character(corp[[3]]))
```

```
Happy with the process, but salary increment in 2019 is very low as
compared to previous years.
```

- **writelines()** prints text line of specified number in [[]]. Here it is printing 3rd line.

Text Mining In R

```
# Clean the Corpus for further analysis
corp <- tm_map(corp, tolower)
writeLines(as.character(corp[[3]]))
```

```
happy with the process, but salary increment in 2019 is very low as
compared to previous years.
```

```
corp <- tm_map(corp, removePunctuation)
writeLines(as.character(corp[[3]]))
```

```
happy with the process but salary increment in 2019 is very low as
compared to previous years
```

```
corp <- tm_map(corp, removeNumbers)
writeLines(as.character(corp[[3]]))
```

```
happy with the process but salary increment in is very low as compared
to previous years
```

- □ **tm_map**() applies transformation functions to a corpus.
- □ **tolower** converts text to lowercase.
- □ **removePunctuation** removes punctuation.
- □ **removeNumbers** removes numbers.

Text Mining In R

```
# Clean the Corpus for further analysis
corp <- tm_map(corp, removeWords, stopwords("english"))
writeLines(as.character(corp[[3]]))
```

```
happy process salary increment low compared previous years
```

```
corp <- tm_map(corp, removeWords, "process")
writeLines(as.character(corp[[3]]))
```

```
happy salary increment low compared previous years
```

- ❑ **removeWords, stopwords("english")** remove stop words like: i, me, our and, the, is, etc. There are more than 100 in-built English Stopwords in R. Use **stopwords("english")** to view the list of these stopwords.
- ❑ If you wish to remove specific words from the corpus, use **tm_map(corp, removeWords, "word")**. Here “**process**” word is removed.

Text Mining In R

```
# Convert to term-document matrix format
```

```
tdm <- TermDocumentMatrix(corp)
findFreqTerms(tdm)
```

```
# Find terms with frequency of at least 5 and find words having high
association with ‘difficult’, ‘questions’
```

```
findFreqTerms(tdm,5)
findAssocs(tdm, 'difficult', 0.60 )
findAssocs(tdm, 'questions', 0.60 )
```

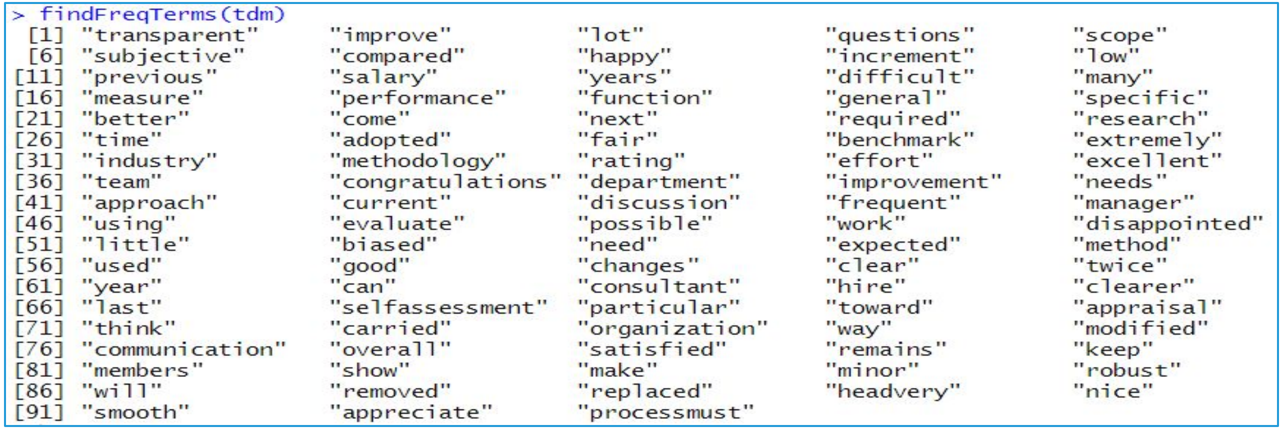
❑ **TermDocumentMatrix()** finds frequent terms in a document-term or term-document matrix. Default minimum frequency is 1 and maximum is infinite. **DocumentTermMatrix()** and **TermDocumentMatrix()** gives the same output.

❑ **findFreqTerms()** gives words with minimum specified frequency . **findFreqTerms**(tdm,5) gives words having minimum frequency 5.

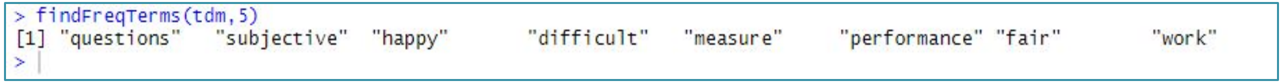
❑ **findAssocs()** gives words with specified minimum correlations with the given word. **findAssocs**(tdm, 'difficult', 0.60) gives words with at least 0.6 correlation with word ‘difficult’.

Text Mining In R

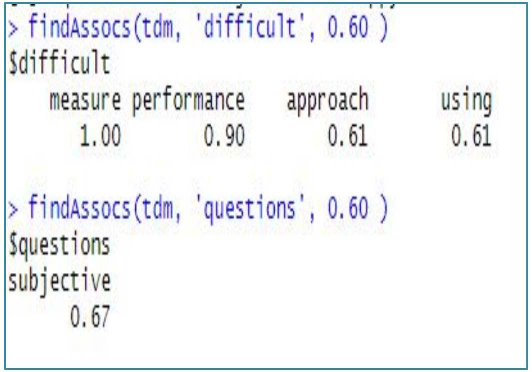
# Output:



```
> findFreqTerms(tdm)
 [1] "transparent"     "improve"          "lot"            "questions"     "scope"
 [6] "subjective"      "compared"         "happy"          "increment"     "low"
[11] "previous"        "salary"           "years"          "difficult"     "many"
[16] "measure"         "performance"      "function"       "general"       "specific"
[21] "better"          "come"             "next"           "required"      "research"
[26] "time"            "adopted"          "fair"           "benchmark"     "extremely"
[31] "industry"        "methodology"      "rating"         "effort"        "excellent"
[36] "team"            "congratulations"  "department"     "improvement"   "needs"
[41] "approach"        "current"          "discussion"     "frequent"      "manager"
[46] "using"           "evaluate"         "possible"       "work"          "disappointed"
[51] "little"          "biased"           "need"           "expected"      "method"
[56] "used"            "good"             "changes"        "clear"         "twice"
[61] "year"            "can"              "consultant"     "hire"          "clearer"
[66] "last"            "selfassessment"   "particular"     "toward"        "appraisal"
[71] "think"           "carried"          "organization"   "way"           "modified"
[76] "communication"   "overall"          "satisfied"      "remains"       "keep"
[81] "members"         "show"             "make"           "minor"         "robust"
[86] "will"            "removed"          "replaced"       "headvery"      "nice"
[91] "smooth"          "appreciate"       "processmust"
```



```
> findFreqTerms(tdm,5)
[1] "questions"   "subjective"  "happy"       "difficult"   "measure"     "performance" "fair"        "work"
> 
```



```
> findAssocs(tdm, 'difficult', 0.60 )
$difficult
    measure performance    approach       using 
       1.00        0.90        0.61        0.61 

> findAssocs(tdm, 'questions', 0.60 )
$questions
subjective 
      0.67 
```

Interpretation:

- questions, subjective, happy, difficult, measure, performance, fair, work are appearing more than 5 times.
- Word ‘difficult’ is having high correlation with measure, performance.

Word Cloud In R

Word cloud, as the name suggests, is an **image showing compilation of words**, in which, **size of words indicates its frequency or importance**.

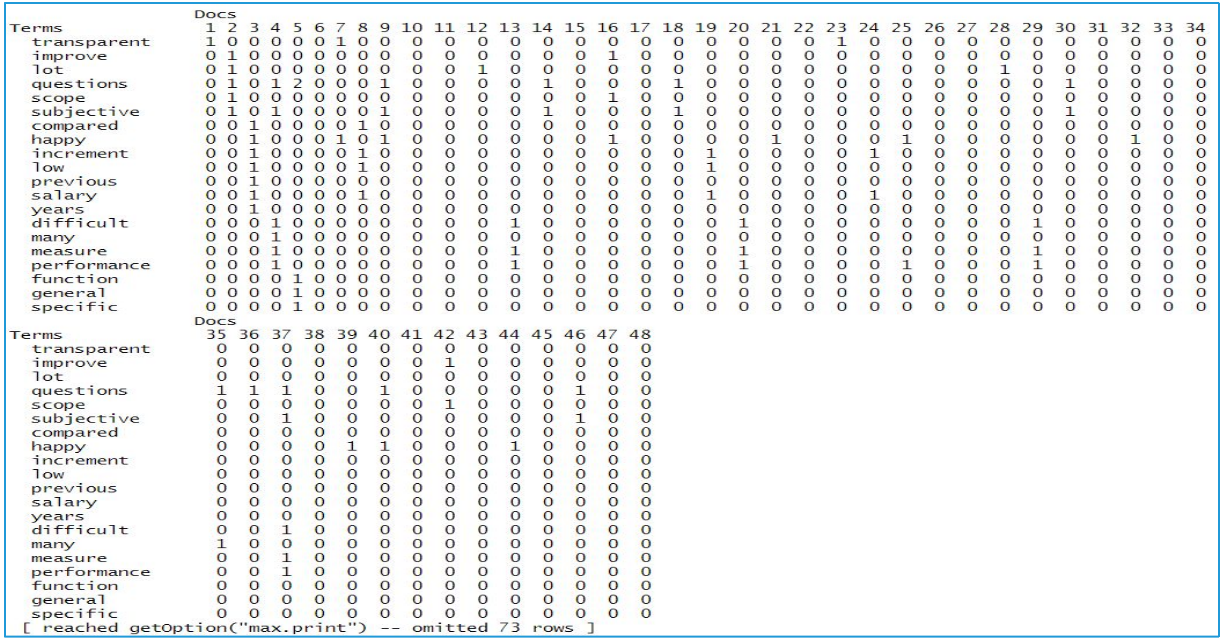
```
# Install and load package “wordcloud”
```

```r
install.packages("wordcloud")
library(wordcloud)
```

```
# Convert tdm object to a matrix
```

```r
m <- as.matrix(tdm)
m
```

Word Cloud In R



```
                 Docs
Terms            1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
  transparent    1 0 0 0 0 0 1 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
  improve        0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  lot            0 1 0 0 0 0 0 0 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
  questions      0 1 0 1 2 0 0 0 1  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
  scope          0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  subjective     0 1 0 1 0 0 0 0 1  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
  compared       0 0 1 0 0 0 0 1 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  happy          0 0 1 0 0 0 1 0 1  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  1  0  0
  increment      0 0 1 0 0 0 0 1 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
  low            0 0 1 0 0 0 0 1 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  previous       0 0 1 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  salary         0 0 1 0 0 0 0 1 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
  years          0 0 1 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  difficult      0 0 0 1 0 0 0 0 0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0
  many           0 0 0 1 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  measure        0 0 0 1 0 0 0 0 0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0
  performance    0 0 0 1 0 0 0 0 0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  1  0  0  0  0  0
  function       0 0 0 0 1 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  general        0 0 0 0 1 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  specific       0 0 0 0 1 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
                 Docs
Terms            35 36 37 38 39 40 41 42 43 44 45 46 47 48
  transparent     0  0  0  0  0  0  0  0  0  0  0  0  0  0
  improve         0  0  0  0  0  0  0  1  0  0  0  0  0  0
  lot             0  0  0  0  0  0  0  0  0  0  0  0  0  0
  questions       1  1  1  0  0  1  0  0  0  0  0  1  0  0
  scope           0  0  0  0  0  0  0  1  0  0  0  0  0  0
  subjective      0  0  1  0  0  0  0  0  0  0  0  1  0  0
  compared        0  0  0  0  0  0  0  0  0  0  0  0  0  0
  happy           0  0  0  0  1  1  0  0  0  1  0  0  0  0
  increment       0  0  0  0  0  0  0  0  0  0  0  0  0  0
  low             0  0  0  0  0  0  0  0  0  0  0  0  0  0
  previous        0  0  0  0  0  0  0  0  0  0  0  0  0  0
  salary          0  0  0  0  0  0  0  0  0  0  0  0  0  0
  years           0  0  0  0  0  0  0  0  0  0  0  0  0  0
  difficult       0  0  1  0  0  0  0  0  0  0  0  0  0  0
  many            1  0  0  0  0  0  0  0  0  0  0  0  0  0
  measure         0  0  1  0  0  0  0  0  0  0  0  0  0  0
  performance     0  0  1  0  0  0  0  0  0  0  0  0  0  0
  function        0  0  0  0  0  0  0  0  0  0  0  0  0  0
  general         0  0  0  0  0  0  0  0  0  0  0  0  0  0
  specific        0  0  0  0  0  0  0  0  0  0  0  0  0  0
 [ reached getOption("max.print") -- omitted 73 rows ]
```

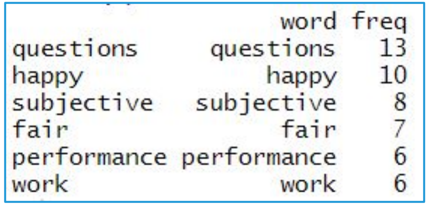
Interpretation:

- There are 48 docs (text lines).
- Example of how to read this output table: Term ‘transparent’ is appearing once in docs 1,7,23 and so on.,

Word Cloud In R

```
# Calculate total frequency of words & creating a data frame of it
```

```
v <- sort(rowSums(m), decreasing=TRUE)
myNames <- names(v)
d <- data.frame(word=myNames, freq=v)
head(d)
```



```
                   word freq
questions     questions   13
happy             happy   10
subjective   subjective    8
fair               fair    7
performance performance    6
work               work    6
```

```
# Create color palette
```

```
pal2 <- brewer.pal(8,"Dark2")
```

❑ **brewer.pal** () was developed by Cynthia Brewer. It makes the color palettes from ColorBrewer available as R palettes.

❑ **Arguments:**

Number of colors included in the palette: 8

Palette Name: 'Dark 2‘

❑ Check out different palettes at **http://colorbrewer2.org/**

Word Cloud In R

```
# Get Word Cloud
```

```
wordcloud(d$word, d$freq, random.order = FALSE , min.freq =
1,colors=pal2)
```

❑ First and second argument in **wordcloud()** are the words **(d$word)** and the frequency **(d$freq)** respectively.

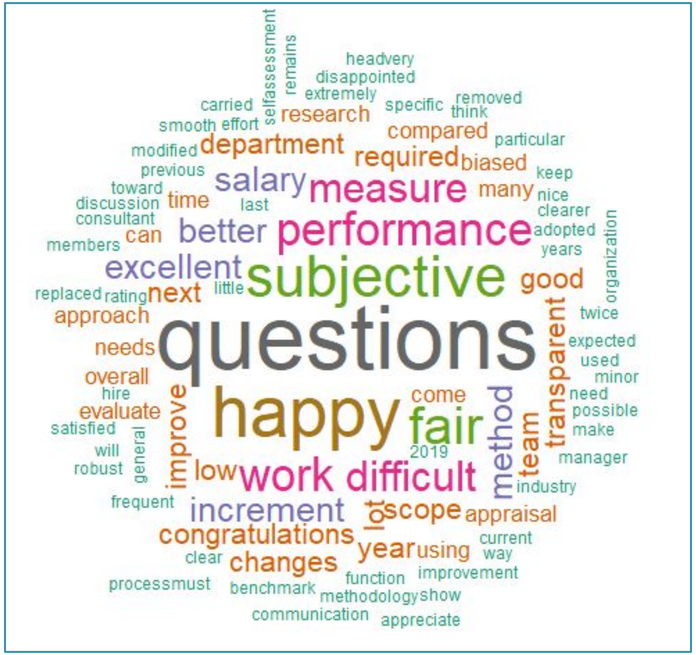
❑ **random.order=FALSE** plots words in decreasing frequency. By default, plot words in random order.

❑ **min.freq =** words with frequency below min.freq will not be plotted.

❑ **colors =** color words from least to most frequent with specified color palette.

Word Cloud In R

```
# Output :
```



Interpretation:
Word questions has the largest size, indicating most frequent word followed by happy and subjective and so on..

Text Mining Using ggplot2

Plotting frequent terms as a bar plot

```
term.freq <- rowSums(m)
term.freq <- subset(term.freq, term.freq >= 5)
```

```
# Transform as a dataframe
```

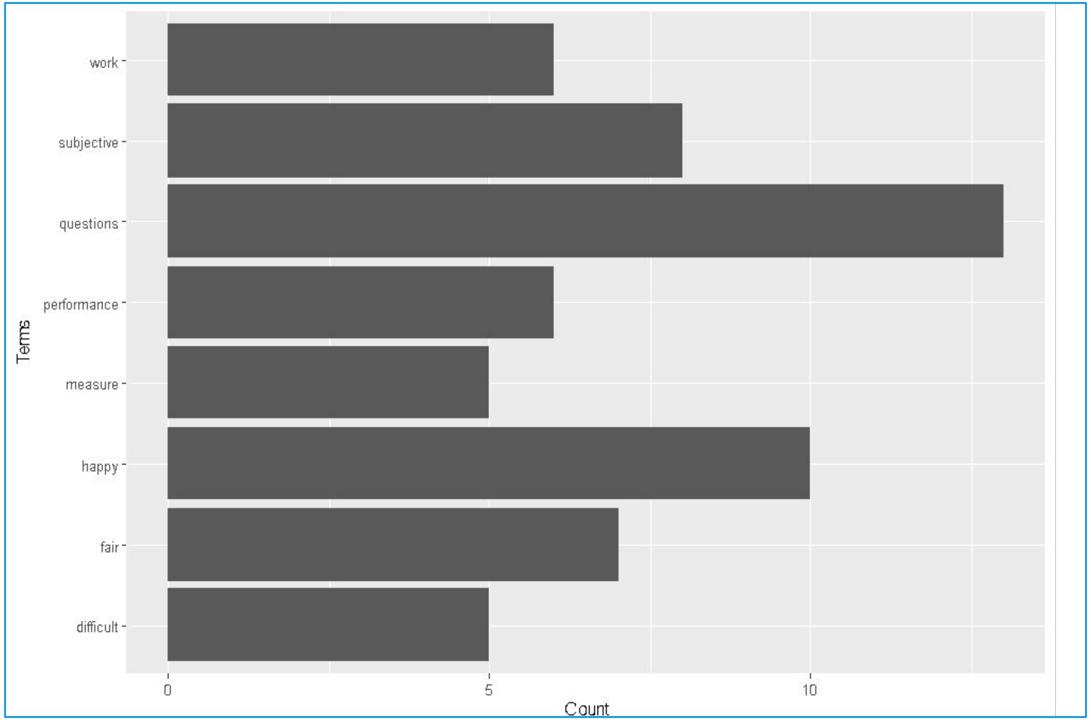
```
df <- data.frame(term = names(term.freq), freq = term.freq)
```

```
# Horizontal bar plot
```

```
install.packages("ggplot2")
library(ggplot2)
ggplot(df, aes(x = term, y = freq))+
  geom_bar(stat = "identity") +
  xlab("Terms") + ylab("Count") + coord_flip()
```

Text Mining Using ggplot2

```
# Output :
```



Interpretation:

- Graph shows the frequency of the words appearing at least 5 times on a horizontal bar graph. questions is the most frequent word with frequency more than 10.

Quick Recap

Unstructured Data

- Does not reside in traditional databases and data warehouses.
- Example: emails, tweets, feedback, blogs, webpages, etc.

Text Analysis

- Process of identifying novel information from a collection of texts. (Also known as a 'Corpus')

Text mining in R

- Install 'tm' package. Convert data into corpus.
- Clean the corpus: convert all to lowercase/uppercase, remove punctuation, numbers, stopwords, words.

Word Cloud in R

- An image showing compilation of words, in which, size of words indicates its frequency or importance.
- Install 'wordcloud' package.

Natural Language Processing - II

Contents

1. What Is NLP
2. Sentiment Analysis
3. Sentiment Analysis using R-Package “sentimentr”
4. Sentiment Analysis using R-Package "syuzhet“

What Is NLP

- Natural Language Processing (NLP) is the ability of a computer to analyze and process natural language data.
- NLP is used to extract relevant information from a piece of text which is then used for various purposes.
- NLP works on four levels - lexical, syntactic, semantic, pragmatic.

  - Lexical - pre-processing of the text, such as removal of stop words, making all text lowercase etc.

  - Syntax analysis - It analyses the 'structure' of text, 'correctness' of a sentence in terms of the grammar of the language of origin.

  - Semantic assessment - attempts to study the 'meaning' of the text.

  - Pragmatic - the analysis is aimed at deciphering the 'intended' meaning of the text.

Applications of NLP

Some of the most notable applications of NLP are:

- Search algorithms - when you search for “What is the population of India”, the top result shows the actual answer
- General websites - Pop-up windows on websites offering ‘chat with their representative’ these are chatbots trained to correctly answer commonly asked questions.
- Retail - Assessment of product feedback using text summarization and sentiment analysis; Query resolution with automated responses.
- Personalized services - Email apps predicting next word(s) in an email, tagging emails as important, personal etc.
- Translation Apps

Sentiment Analysis

- Sentiment Analysis is the **process of determining whether a piece of writing is positive, negative or neutral**.
- It's also known as opinion mining, deriving the opinion or attitude of a speaker.
- Sentiment analysis is performed using natural language processing, text analysis, computational linguistics and, sometimes, biometrics to systematically identify, extract, quantify, and study affective states and subjective information.
- Basic task in sentiment analysis is classifying the **polarity** of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion is positive, negative, or neutral.
- Advanced - "beyond polarity" - sentiment classification looks at emotional states, for instance, "angry", "sad", and "happy".

Sentiment Analysis Using "sentimentr"

- The sentimentr package was developed by Tyler Rinker.

- It adopts a dictionary lookup approach that tries to incorporate weighting for valence shifters.

- It's aim is to improve the polarity recognition performance with respect to the syuzhet package (which does not recognize valence shifters), it does so at the expense of speed.

Sentiment Analysis Using "sentimentr"

```
# Install and Load package “sentimentr”
```

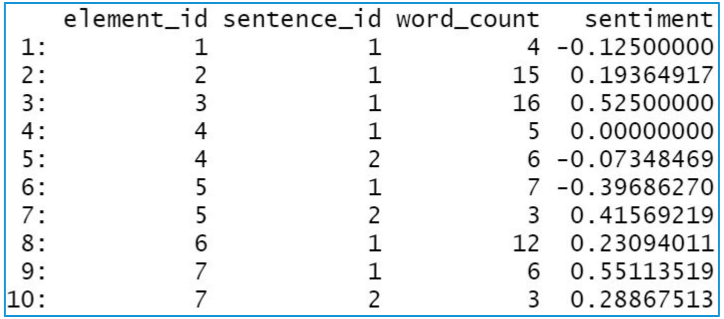
```
install.packages("sentimentr")
library(sentimentr)
```

```
# Calculate Sentiment Score
```

```
data<-readLines("HR Appraisal process.txt")
sentiment(data)
```

- ❑ **sentiment()** calculates the sentiment values of each sentence in the data.

```
# Output
```



```
    element_id sentence_id word_count   sentiment
 1:          1           1          4 -0.12500000
 2:          2           1         15  0.19364917
 3:          3           1         16  0.52500000
 4:          4           1          5  0.00000000
 5:          4           2          6 -0.07348469
 6:          5           1          7 -0.39686270
 7:          5           2          3  0.41569219
 8:          6           1         12  0.23094011
 9:          7           1          6  0.55113519
10:          7           2          3  0.28867513
```

- ❑ **element_id** is the id number of the original vector passed to sentiment
- ❑ **sentence_id** is the id number of the sentences within each element_id
- ❑ **word_count** is the count of words in each sentence
- ❑ **sentiment** is the sentiment/polarity score of each sentence

Sentiment Analysis Using "sentimentr"

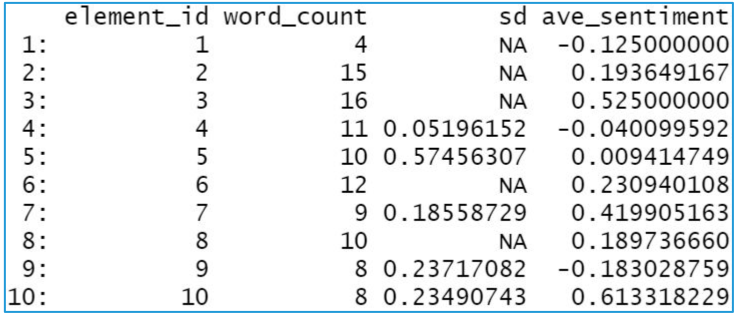
```
# Aggregate sentiment scores”
```

You can also calculate the sentiment scores by aggregating it with respect to different elements. The default value is by=”NULL”, which aggregates the sentiment scores with respect to each line

```
# Calculate Avg Sentiment Score
sentiment_by(data)
```

❑ **Sentiment_by()** calculates the aggregate sentiment values

```
# Output
```



```
   element_id word_count         sd ave_sentiment
 1:          1          4         NA  -0.125000000
 2:          2         15         NA   0.193649167
 3:          3         16         NA   0.525000000
 4:          4         11 0.05196152  -0.040099592
 5:          5         10 0.57456307   0.009414749
 6:          6         12         NA   0.230940108
 7:          7          9 0.18558729   0.419905163
 8:          8         10         NA   0.189736660
 9:          9          8 0.23717082  -0.183028759
10:         10          8 0.23490743   0.613318229
```

❑ **sd** gives the standard deviation of the sentiment score of the sentences in the review

❑ **ave_sentiment** gives the average sentiment score of the sentences in the review

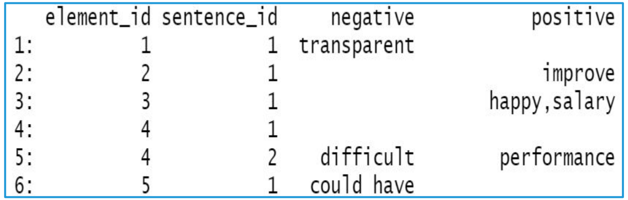
* Note : Sentiment Categories can be derived using Sentiment Scores

Sentiment Analysis Using "sentimentr"

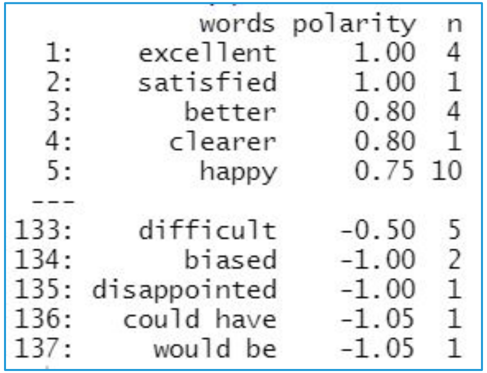
```
# Extract Sentiment words and polarity form each sentence
t<-extract_sentiment_terms(data)
head(t)
attributes(t)$count
```

- **extract_sentiment_terms()** extracts the sentiment words from a text.
- **attributes(t)$count** return an aggregated count of the usage of the words and a detailed sentiment score of each word use.

```
# Output
```



```
   element_id sentence_id    negative     positive
1:          1           1 transparent
2:          2           1                  improve
3:          3           1             happy,salary
4:          4           1
5:          4           2   difficult  performance
6:          5           1  could have
```



```
             words polarity  n
  1:     excellent     1.00  4
  2:     satisfied     1.00  1
  3:        better     0.80  4
  4:       clearer     0.80  1
  5:         happy     0.75 10
 ---
133:     difficult    -0.50  5
134:        biased    -1.00  2
135:  disappointed    -1.00  1
136:    could have    -1.05  1
137:      would be    -1.05  1
```

Sentiment Analysis Using "syuzhet"

- Syuzhet is an R package for the extraction of sentiment and sentiment-based plot arcs from text.
- The name “Syuzhet” comes from the Russian Formalists Victor Shklovsky and Vladimir Propp who divided narrative into two components, the “fabula” and the “syuzhet.” Syuzhet refers to the “device” or technique of a narrative whereas fabula is the chronological order of events. Syuzhet, therefore, is concerned with the manner in which the elements of the story (fabula) are organized (syuzhet).
- The package is more suitable for analysis of sentiment trajectory across a text document.

Sentiment Analysis Using "syuzhet"

- Syuzhet incorporates four sentiment lexicons:

“syuzhet” (Default)

- Developed in the Nebraska Literary Lab under the direction of Matthew L. Jockers

“afinn”

- Developed by Finn Arup Nielsen as the AFINN WORD DATABASE

“bing”

- Developed by Minqing Hu and Bing Liu as the OPINION LEXICON

“nrc”

- Developed by Mohammad, Saif M. and Turney, Peter D. as the NRC EMOTION LEXICON

Sentiment Analysis Using "syuzhet"

```
# Install and Load package “syuzhet”
```

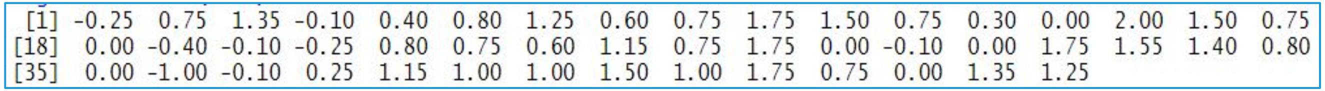
```
install.packages("syuzhet")
library(syuzhet)
```

```
# Calculate Sentiment Values
```

```
get_sentiment(data)
```

- ❑ **get_sentiment()** calculates sentiment of each word or sentence. First argument is a character vector (or sentences or words) and second argument is for method (Which lexicon to be used). The function uses **method="syuzhet"** by default.
- ❑ We have passed data object as it is to the function. This ensures feedback with more than one sentences is not split and considered entirely. However, it is a better practice to 'tokenise' data. Sentences separated by full stops are split.

```
# Output
```



```
 [1] -0.25  0.75  1.35 -0.10  0.40  0.80  1.25  0.60  0.75  1.75  1.50  0.75  0.30  0.00  2.00  1.50  0.75
[18]  0.00 -0.40 -0.10 -0.25  0.80  0.75  0.60  1.15  0.75  1.75  0.00 -0.10  0.00  1.75  1.55  1.40  0.80
[35]  0.00 -1.00 -0.10  0.25  1.15  1.00  1.00  1.50  1.00  1.75  0.75  0.00  1.35  1.25
```

Sentiment Analysis Using "syuzhet"

```
# Display emotions and valence from NRC dictionary
```

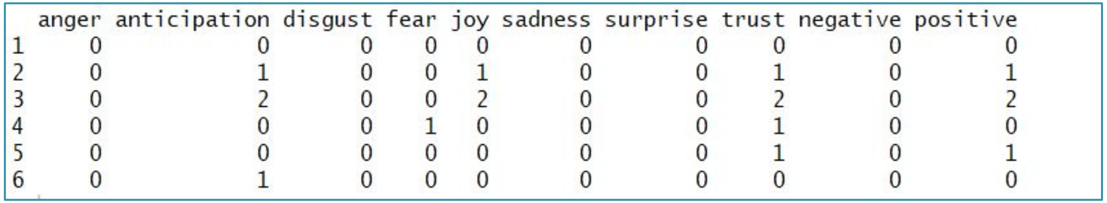
```
nrcsentiment <- get_nrc_sentiment(data)
head(nrcsentiment)
```

❑ get_nrc_sentiment() calls the NRC sentiment dictionary to calculate the presence of eight different emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust)and two sentiments (positive and negative).

❑ It returns a data frame in which each row represents a sentence from the original file. The columns include one for each emotion type as well as the positive or negative sentiment valence.

Sentiment Analysis Using "syuzhet"

```
# Output :
```



```
  anger anticipation disgust fear joy sadness surprise trust negative positive
1     0            0       0    0   0       0        0     0        0        0
2     0            1       0    0   1       0        0     1        0        1
3     0            2       0    0   2       0        0     2        0        2
4     0            0       0    1   0       0        0     1        0        0
5     0            0       0    0   0       0        0     1        0        1
6     0            1       0    0   0       0        0     0        0        0
```

Interpretation:

- Negative score indicates ,negative sentiments.
- Example of how to read output table : second sentence is having joyful sentiments. 4th sentence has fear sentiment.

Sentiment Plot Using "syuzhet"

- It is sometimes useful to plot the values in a graph where the x-axis represents the passage of time from the beginning to the end of the text, and the y-axis measures the degrees of positive and negative sentiment.

- **`plot()`** is a function used for plotting the sequence of sentences in terms of emotions and sentiments.

- This works best when the text being analysed is part of a single document (Like a book, novel, essay, etc.)

Sentiment Plot Using "syuzhet"

```
# Plot the sentiments across time
```

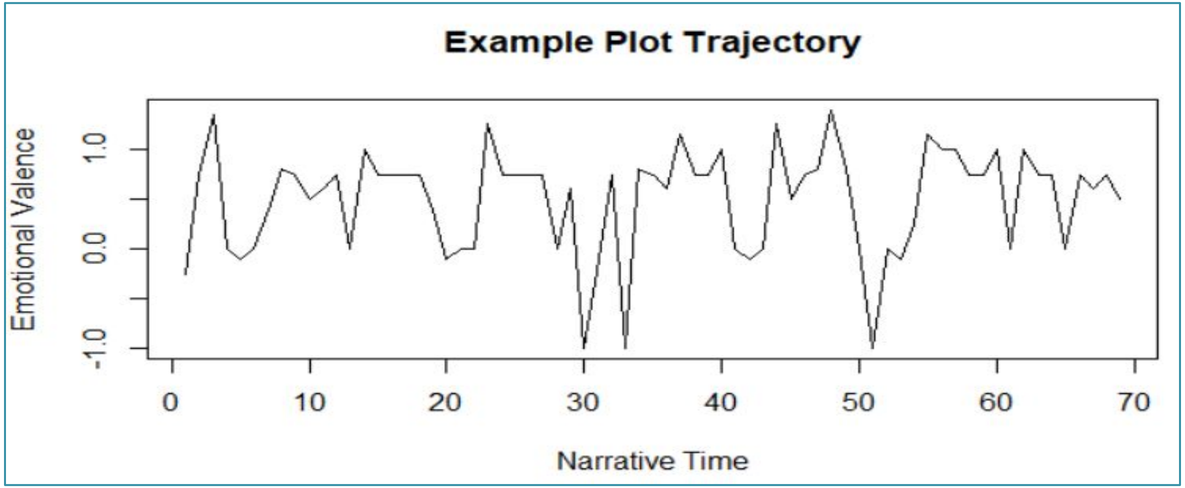
```
read_sentence<- get_sentences(data)
read_sentence_sentiment <- get_sentiment(read_sentence)

plot(
  read_sentence_sentiment,
  type="l",
  main="Example Plot Trajectory",
  xlab = "Narrative Time",
  ylab= "Emotional Valence"
)
```

- ❑ **get_sentences()** parses a string into a vector of sentences
- ❑ **get_sentiment()** calculates sentiment of each word or sentence. First argument is a character vector (or sentences or words) and second argument is for method (Which lexicon to be used). The function uses method="syuzhet“ by default.
- ❑ **type=“1”** is for lines

Sentiment Plot Using "syuzhet"

```
# Output :
```



Interpretation:

- Data mostly varies between neutral and positive sentiments with three negative streaks.

Quick Recap

In this session, we learnt about **NLP & Sentiment Analysis** :

NLP

- Natural Language Processing (NLP) is the ability of a computer to analyze and process natural language data.

Sentiment Analysis

- Sentiment Analysis is the process of determining whether a piece of writing is positive, negative or neutral.
- 'sentimentr' & 'syuzhet' packages can be used for sentiment analysis.