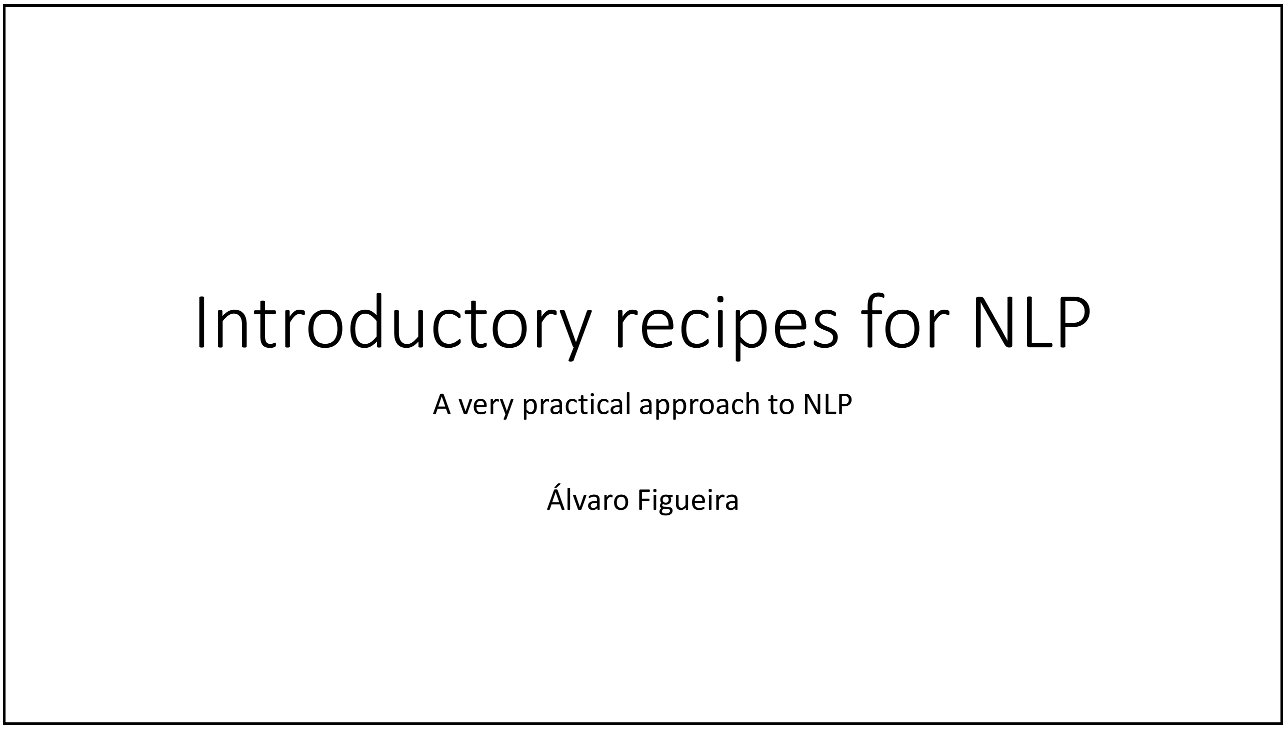
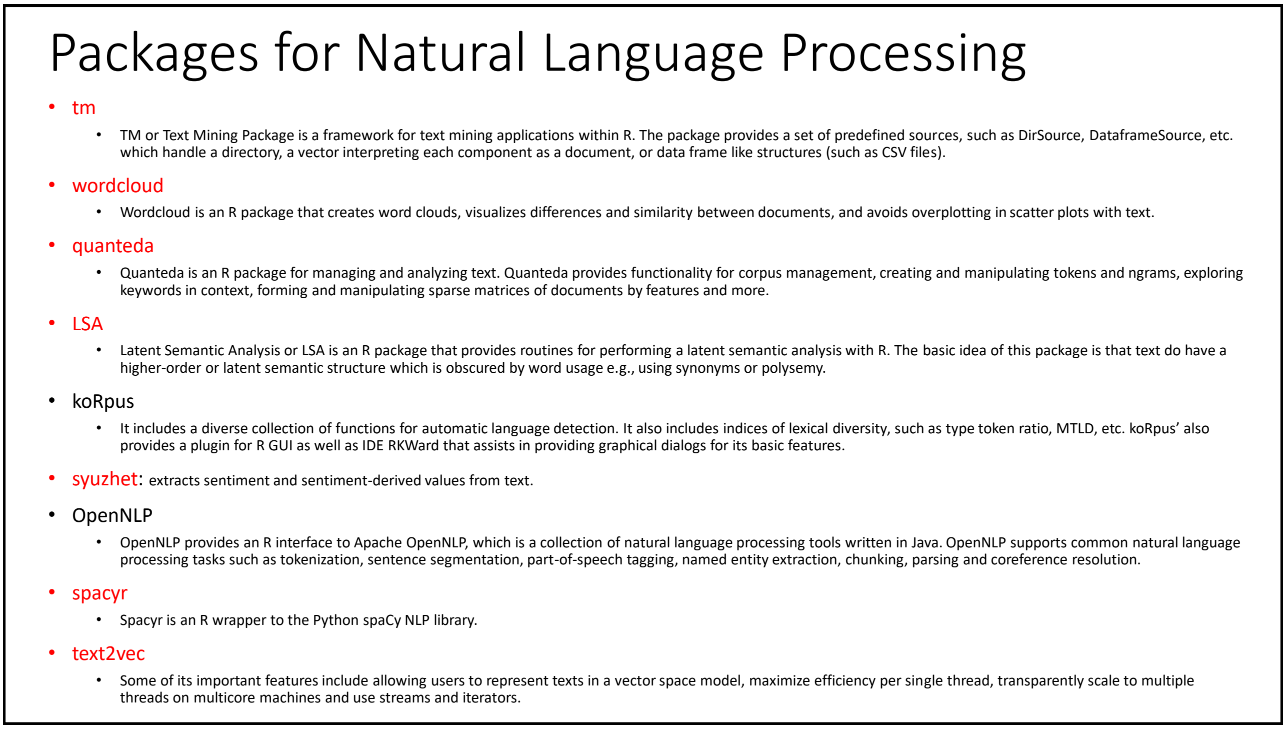


Introductory recipes for NLP

A very practical approach to NLP

Álvaro Figueira

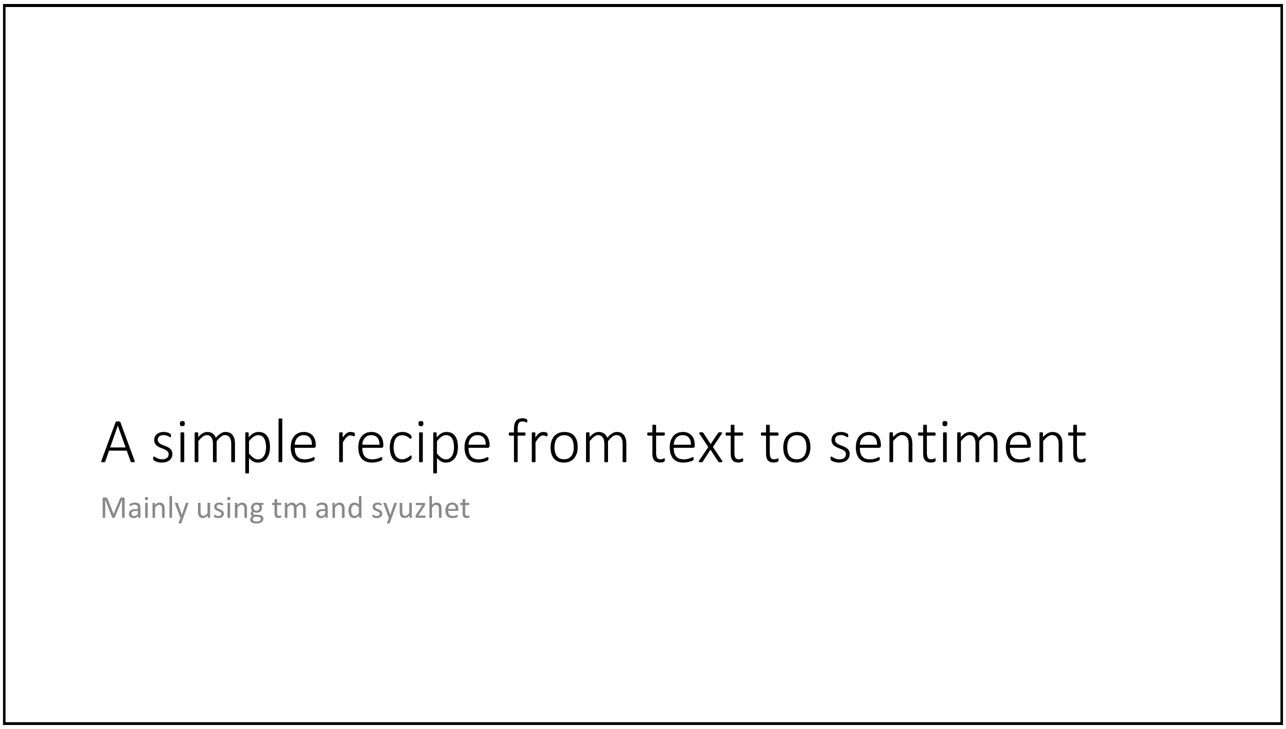
1



Packages for Natural Language Processing

- tm
  - TM or Text Mining Package is a framework for text mining applications within R. The package provides a set of predefined sources, such as DirSource, DataframeSource, etc. which handle a directory, a vector interpreting each component as a document, or data frame like structures (such as CSV files).
- wordcloud
  - Wordcloud is an R package that creates word clouds, visualizes differences and similarity between documents, and avoids overplotting in scatter plots with text.
- quanteda
  - Quanteda is an R package for managing and analyzing text. Quanteda provides functionality for corpus management, creating and manipulating tokens and ngrams, exploring keywords in context, forming and manipulating sparse matrices of documents by features and more.
- LSA
  - Latent Semantic Analysis or LSA is an R package that provides routines for performing a latent semantic analysis with R. The basic idea of this package is that text do have a higher-order or latent semantic structure which is obscured by word usage e.g., using synonyms or polysemy.
- koRpus
  - It includes a diverse collection of functions for automatic language detection. It also includes indices of lexical diversity, such as type token ratio, MTLD, etc. koRpus' also provides a plugin for R GUI as well as IDE RKWard that assists in providing graphical dialogs for its basic features.
- syuzhet: extracts sentiment and sentiment-derived values from text.
- OpenNLP
  - OpenNLP provides an R interface to Apache OpenNLP, which is a collection of natural language processing tools written in Java. OpenNLP supports common natural language processing tasks such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing and coreference resolution.
- spacyr
  - Spacyr is an R wrapper to the Python spaCy NLP library.
- text2vec
  - Some of its important features include allowing users to represent texts in a vector space model, maximize efficiency per single thread, transparently scale to multiple threads on multicore machines and use streams and iterators.

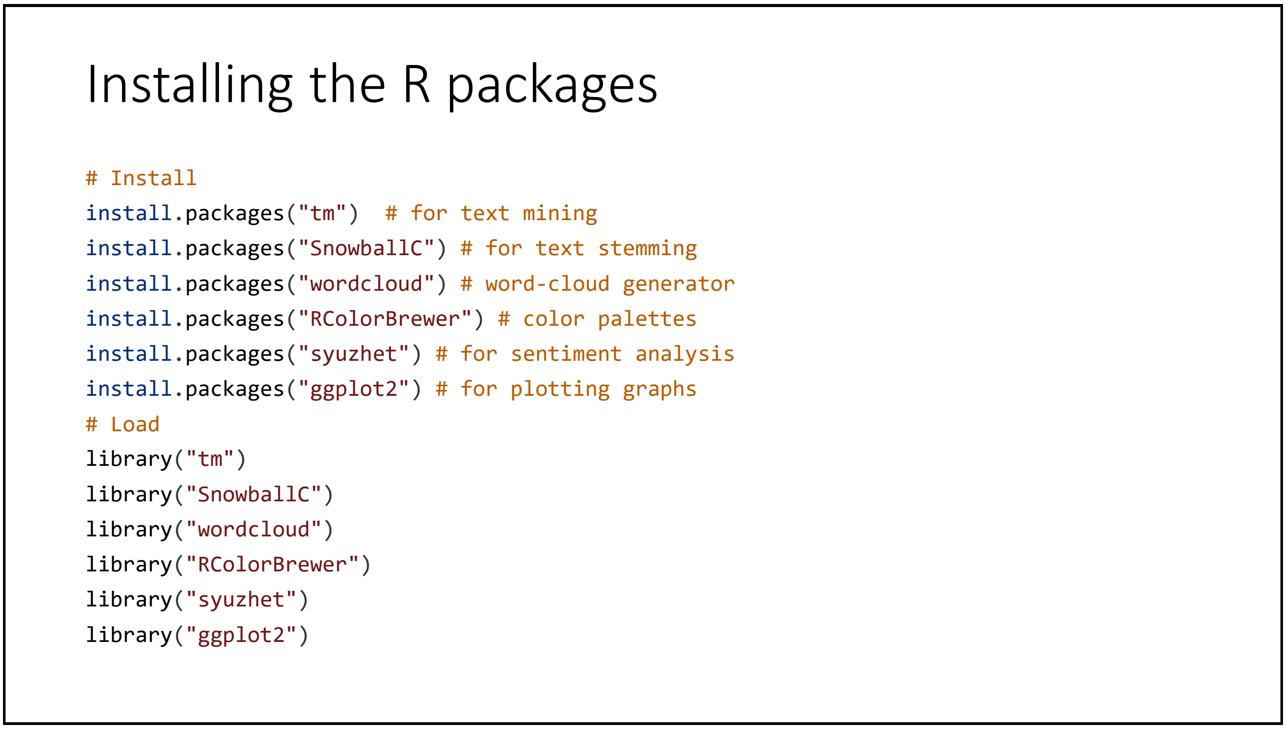
2



A simple recipe from text to sentiment

Mainly using tm and syuzhet

8



Installing the R packages

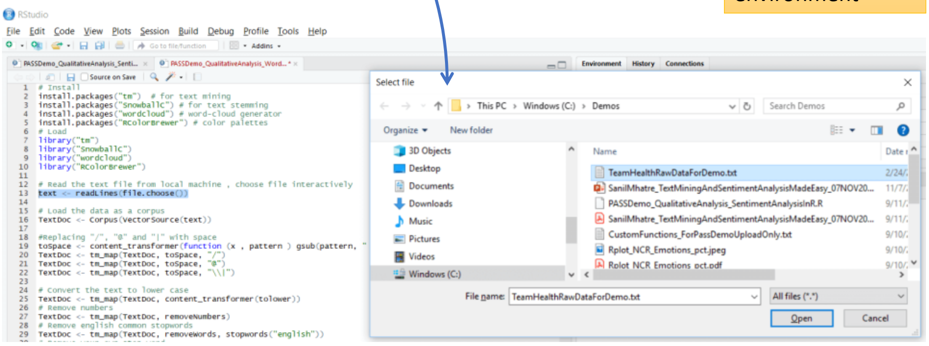
```r
# Install
install.packages("tm")  # for text mining
install.packages("SnowballC") # for text stemming
install.packages("wordcloud") # word-cloud generator
install.packages("RColorBrewer") # color palettes
install.packages("syuzhet") # for sentiment analysis
install.packages("ggplot2") # for plotting graphs
# Load
library("tm")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")
library("syuzhet")
library("ggplot2")
```

9

Reading file data into R

- # Read the text file from local machine , choose file interactively
- text <- readLines(file.choose())
- # Load the data as a corpus
- TextDoc <- Corpus(VectorSource(text))

Suppose a questionnaire about the working environment


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
PASSDemo_QualitativeAnalysis_Senti...
PASSDemo_QualitativeAnalysis_Word...*
Source on Save
Environment
History
Connections
Select file
This PC › Windows (C:) › Demos
Search Demos
Organize
New folder
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Name
Date r
TeamHealthRawDataForDemo.txt
SanilMhatre_TextMiningAndSentimentAnalysisMadeEasy_07NOV20...
PASSDemo_QualitativeAnalysis_SentimentAnalysisInR.R
SanilMhatre_TextMiningAndSentimentAnalysisMadeEasy_07NOV20...
CustomFunctions_ForPassDemoUploadOnly.txt
Rplot_NCR_Emotions_pct.jpeg
Rplot_NCR_Emotions_pct.pdf
All files (*.*)
Open
Cancel

```
# Install
install.packages("tm")  # for text mining
install.packages("SnowballC") # for text stemming
install.packages("wordcloud") # word-cloud generator
install.packages("RColorBrewer") # color palettes
# Load
library("tm")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")

# Read the text file from local machine , choose file interactively
text <- readLines(file.choose())

# Load the data as a corpus
TextDoc <- Corpus(VectorSource(text))

#Replacing "/", "@" and "|" with space
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, "
TextDoc <- tm_map(TextDoc, toSpace, "/")
TextDoc <- tm_map(TextDoc, toSpace, "@")
TextDoc <- tm_map(TextDoc, toSpace, "\\|")

# Convert the text to lower case
TextDoc <- tm_map(TextDoc, content_transformer(tolower))
# Remove numbers
TextDoc <- tm_map(TextDoc, removeNumbers)
# Remove english common stopwords
TextDoc <- tm_map(TextDoc, removeWords, stopwords("english"))
```

File name: TeamHealthRawDataForDemo.txt

Cleaning up Text Data

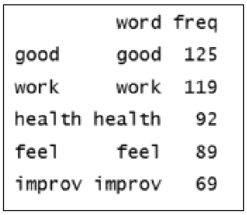
```
# Replacing "/", "@" and "|" with space
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))

TextDoc <- tm_map(TextDoc, toSpace, "/")
TextDoc <- tm_map(TextDoc, toSpace, "@")
TextDoc <- tm_map(TextDoc, toSpace, "\\|")
```

be careful with this

```
# Convert the text to lower case
TextDoc <- tm_map(TextDoc, content_transformer(tolower))
# Remove numbers
TextDoc <- tm_map(TextDoc, removeNumbers)
# Remove english common stopwords
TextDoc <- tm_map(TextDoc, removeWords, stopwords("english"))
# specify your custom stopwords as a character vector
TextDoc <- tm_map(TextDoc, removeWords, c("s", "company", "team"))
# Remove punctuations
TextDoc <- tm_map(TextDoc, removePunctuation)
# Eliminate extra white spaces
TextDoc <- tm_map(TextDoc, stripWhitespace)
# Perform lemmatization/stemming
TextDoc <- tm_map(TextDoc, content_transformer(lemmatize_strings))
TextDoc <- tm_map(TextDoc, stemDocument)
```

Building the Term Document Matrix

```
# Build a term-document matrix
TextDoc_dtm <- TermDocumentMatrix(TextDoc)
dtm_m <- as.matrix(TextDoc_dtm)
# Sort by decreasing value of frequency
dtm_v <- sort(rowSums(dtm_m),decreasing=TRUE)
dtm_d <- data.frame(word = names(dtm_v),freq=dtm_v)
# Display the top 5 most frequent words
head(dtm_d, 5)
```



```
         word freq
good     good  125
work     work  119
health health   92
feel     feel   89
improv improv   69
```

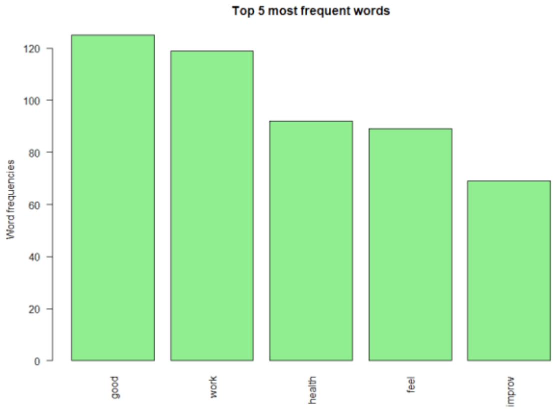
12

The most frequent words

```
# Plot the most frequent words
barplot(dtm_d[1:5,]$freq, las = 2, names.arg = dtm_d[1:5,]$word,
        col ="lightgreen", main ="Top 5 most frequent words",
        ylab = "Word frequencies")
```

One could interpret the following from this bar chart:

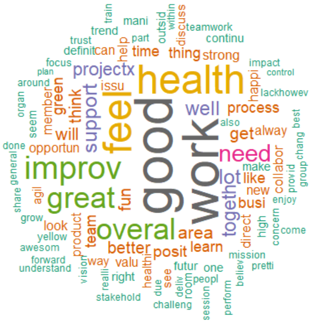
- The most frequently occurring word is “good”. Also notice that negative words like “not” don’t feature in the bar chart, which indicates there are no negative prefixes to change the context or meaning of the word “good” (In short, this indicates most responses don’t mention negative phrases like “not good”).
- “work”, “health” and “feel” are the next three most frequently occurring words, which indicate that most people feel good about their work and their team’s health.
- Finally, the root “improv” for words like “improve”, “improvement”, “improving”, etc. is also on the chart, and you need further analysis to infer if its context is positive or negative.



13

Generate a Word Cloud

```
# Generate word cloud
set.seed(1234)
wordcloud(words = dtm_d$word, freq = dtm_d$freq, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```

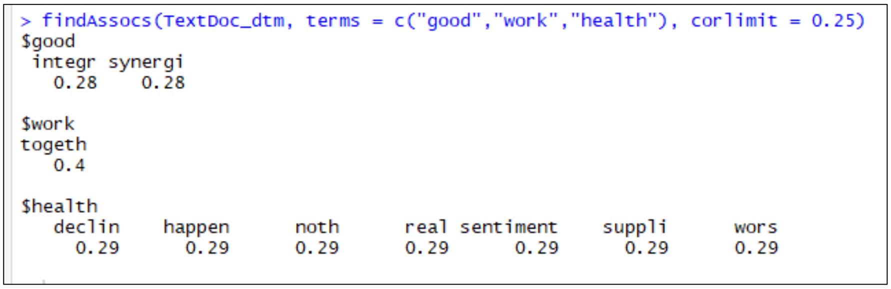


A brief description of the arguments used in the word cloud function:

- **words** – words to be plotted
- **freq** – frequencies of words
- **min.freq** – words whose frequency is at or above this threshold value is plotted
- **max.words** – the maximum number of words to display on the plot
- **random.order** – It is set to FALSE, so the words are plotted in order of decreasing frequency
- **rot.per** – the percentage of words that are displayed as vertical text (with 90-degree rotation).
- **colors** – changes word colors going from lowest to highest frequencies

Word Association (I)

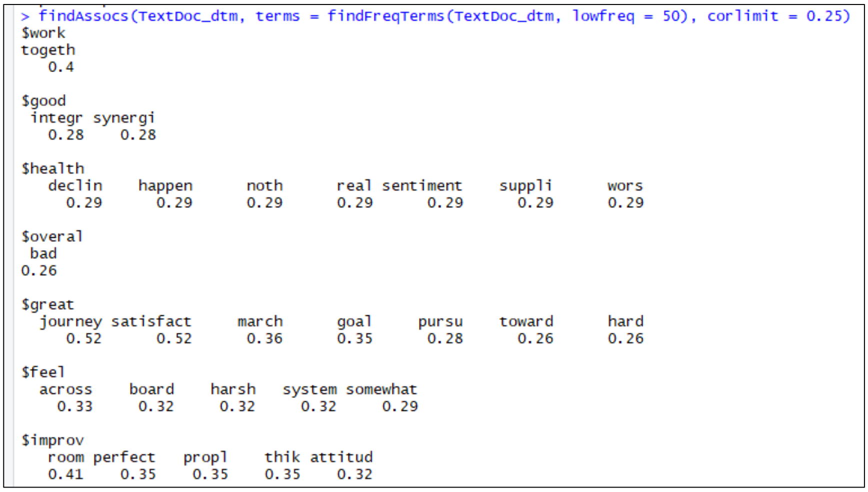
```
# Find associations with above minimum correlation
findAssocs(TextDoc_dtm, terms = c("good","work","health"), corlimit = 0.25)
```



```
> findAssocs(TextDoc_dtm, terms = c("good","work","health"), corlimit = 0.25)
$good
 integr synergi
   0.28    0.28

$work
togeth
   0.4

$health
   declin    happen      noth      real sentiment    suppli      wors
     0.29      0.29      0.29      0.29      0.29      0.29      0.29
```

Word Association (II)

```
# Find associations for words that occur at least 50 times
findAssocs(TextDoc_dtm, terms = findFreqTerms(TextDoc_dtm, lowfreq = 50), corlimit = 0.25)
```

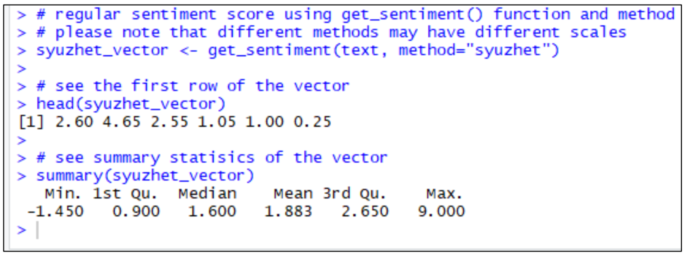


```
> findAssocs(TextDoc_dtm, terms = findFreqTerms(TextDoc_dtm, lowfreq = 50), corlimit = 0.25)
$work
togeth
   0.4

$good
 integr synergi
   0.28    0.28

$health
   declin    happen      noth      real sentiment    suppli      wors
     0.29      0.29      0.29      0.29      0.29      0.29      0.29

$overal
 bad
0.26

$great
  journey satisfact     march      goal     pursu    toward      hard
     0.52      0.52      0.36      0.35      0.28      0.26      0.26

$feel
  across    board    harsh   system somewhat
    0.33     0.32     0.32     0.32     0.29

$improv
   room perfect   propl    thik attitud
   0.41    0.35    0.35    0.35    0.32
```

16

Sentiment Scores

```
# regular sentiment score using get_sentiment() function and one method
# please note that different methods may have different scales
syuzhet_vector <- get_sentiment(text, method="syuzhet")
# see the first row of the vector
head(syuzhet_vector)
# see summary statistics of the vector
summary(syuzhet_vector)
```



```
> # regular sentiment score using get_sentiment() function and method
> # please note that different methods may have different scales
> syuzhet_vector <- get_sentiment(text, method="syuzhet")
>
> # see the first row of the vector
> head(syuzhet_vector)
[1] 2.60 4.65 2.55 1.05 1.00 0.25
>
> # see summary statistics of the vector
> summary(syuzhet_vector)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -1.450   0.900   1.600   1.883   2.650   9.000
>
```

Scales for each lexicon:

- Syuzhet: decimal, negative to positive
- Bing: binary, -1 and 1
- Afinn: integer, -5 to +5
- Vader: integer, -4 to +4

17