TEXT MINING

Using the software R

TO START

First of all, you need to install the packages (or libraries) required for the correct execution of the textual analysis in R:

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

```
install.packages("ggthemes")
install.packages("qdap")
install.packages("dplyr")
install.packages("wordcloud")
install.packages("plotrix")
install.packages("dendextend")
install.packages("ggplot2")
install.packages("ggthemes")
install.packages("RWeka")
install.packages("reshape2")
install.packages("quanteda")
install.packages(plotrix)
install.packages("BiocManager")
BiocManager::install("Rgraphviz")
install.packages("graph")
```

TO START

The downloaded packages must then be recalled:

```
library(qdap)
library(dplyr)
library(wordcloud)
library(plotrix)
library(dendextend)
library(ggplot2)
library(ggthemes)
library(RWeka)
library(reshape2)
library(quanteda)
library(tm)
library(graph)
library(Rgraphviz)
library(plotrix)
```

CORPUS

Load the text file to analyse: «comments»

```
comments <- read_excel("~/Desktop/comments.xlsx")
```

First of all, it is necessary to convert the set of documents in a "Corpus".
To create a corpus using tm, we need to pass a "Source" object as a parameter to the VCorpus method.
The source we use here is a "Vectorsource" which inserts only character vectors.

The "comments" column is now converted to a corpus named "corpus_review".

Command to create a corpus

```
corpus_review=Corpus(VectorSource(comments))
```

Pre-processing: general functions

```
#To inspect the Corpus:
inspect(corpus_review)
#more in details:
writeLines(as.character(corpus_review[1]))

#From uppercase to lowercase:
corpus_review=tm_map(corpus_review, tolower)

#To remove puntuaction:
corpus_review=tm_map(corpus_review, removePunctuation)

#To remove special characters
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
corpus_review= tm_map(corpus_review, toSpace, "\\|")
corpus_review= tm_map(corpus_review, toSpace, " piã'")
corpus_review= tm_map(corpus_review, toSpace, " í ½í¸ˆí ½í¸ˆí ½í¸ˆ í ½í¸¢)
#To remove numbers
   corpus_review=tm_map(corpus_review, removeNumbers)
```

Pre-processing: remove terms in a given language

```
#To remove Italian or English terms
corpus_review=tm_map(corpus_review, removeWords, stopwords("english"))

corpus_review=tm_map(corpus_review, removeWords, stopwords("italian"))
```

Pre-processing: customization to remove specific words

```
#To see how is changed the document:
writeLines(as.character(corpus_review[1]))


#To customize specific words to remove:
corpus_review=tm_map(corpus_review,removeWords,c(“even",“for",“the",“a",
“hello"))
```

Pre-processing: combine words

```
#To combine words that should fit together but which are

separated in the document


for (j in seq(corpus_review))
{corpus_review [[j]] <- gsub("less heat","less_heat"", corpus_review [[j]])
corpus_review [[j]] <- gsub("long hair", "long_hair", corpus_review [[j]]) }


corpus_review <- tm_map(corpus_review, PlainTextDocument)
```

STEMMING

To identify, starting from any morphological variation, the word of origin, stemming programs are usually defined as stemming algorithms or stemmers.

The construction of a stemming algorithm is one of the most complicated steps of the whole process, as it strongly depends on the context in which you are working and above all on the language used.

```
corpus_review=tm_map(corpus_review, stemDocument)
```

DOCUMENT MATRIX

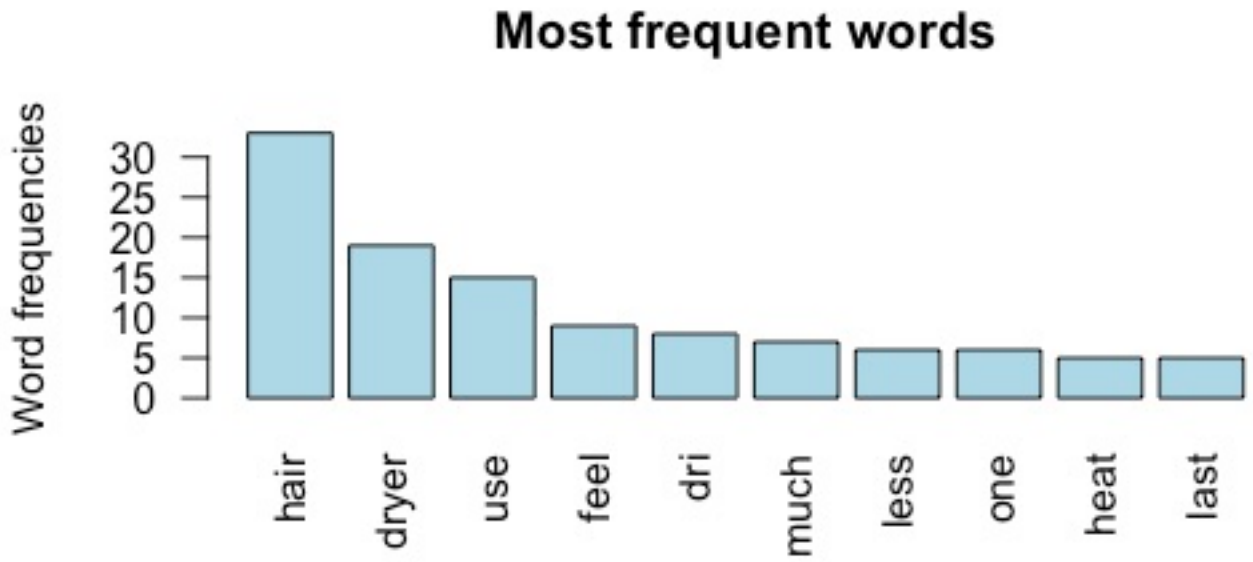
La Document matrix è una tabella contenente le frequenze delle parole.
La funzione TermDocumentMatrix () dal pacchetto R di text mining è la seguente:

```
dtm <- TermDocumentMatrix(corpus_review)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
```

Term	Frequency
Bad	77
Beautiful	58
Need	30
Assistance	26
break	24
small	20
negative	18
failure	10
problem	7
price	5

Word frequency: Bar-plot

```
barplot(d[1:10,]$freq, las = 2, names.arg = d[1:10,]$word, col ="lightblue", main
="Most frequent words", ylab = "Word frequencies")
```



WORD-CLOUD

```
#Command to create a word cloud
set.seed(1234)
wordcloud(words = d$word, freq = d$freq, min.freq = 2,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

Word association

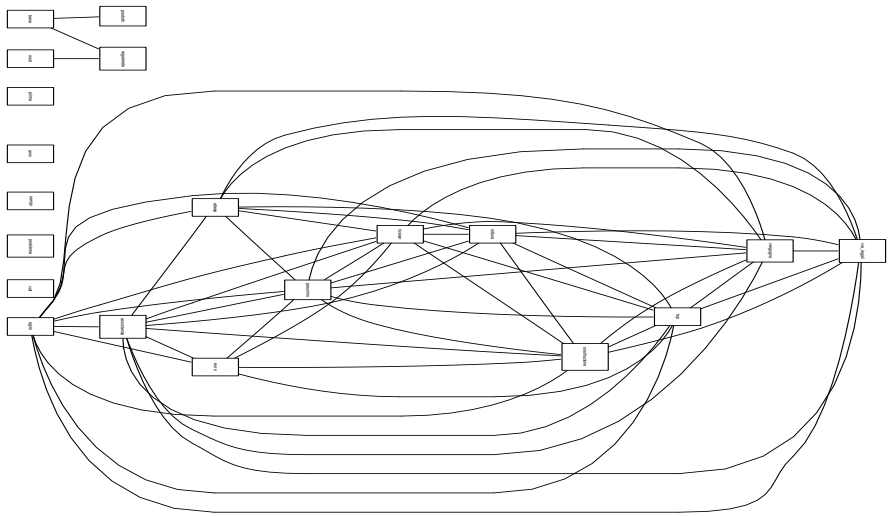
```
dtm <- TermDocumentMatrix(corpus_review)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)

findAssocs(dtm, terms = findFreqTerms(dtm, lowfreq = 50), corlimit = 0.25)
freq.term<-findFreqTerms(dtm, lowfreq=10)
```

#network of terms with an association greater than 0.5

```
plot(dtm, term=freq.term, corThreshold=0.5)
```

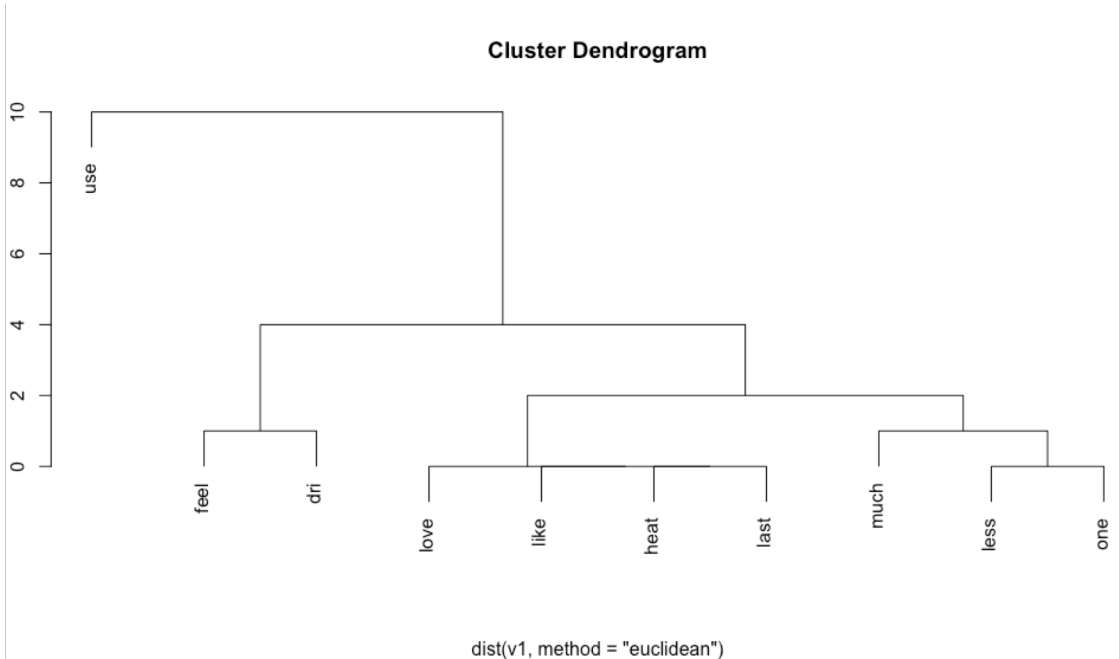
#word-association with an association greater than 0.3 with "break“

```
findAssocs(dtm, «break", 0.3)
```



WORD CLUSTERING

```
dtm <- TermDocumentMatrix(corpus_review)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v) #word cluster
v=sort(v, decreasing=TRUE)
v1=v[v>4]
hc<-hclust(d=dist(v1, method="euclidean"), method="complete")
plot(hc)
```



Polaryzed tag plot (or pyramid plot)

Import two different Excel files that group "comments" on facebook and "private_ messages" **to detect the frequency associated with terms common to the two datasets**

```
comments <- read_excel("dyson comments.xlsx", col_names = FALSE)
negative_comments <- read_excel("dyson comments.xlsx",
                                sheet = "negative", col_names = FALSE)

#create a useful function to delete non-essential terms (as seen above) - the clean.vec function is shown in the R file
com.vec<-clean.vec(comments)
neg.vec<-clean.vec(negative_comments)
```

Polaryzed tag plot

```
#Create a single corpus from the two initial documents cleaned of non-essential terms
com.vec <- paste(com.vec, collapse=" ")
neg.vec <- paste(neg.vec, collapse=" ")
all <- c(com.vec, neg.vec)
corpus <- VCorpus(VectorSource(all))

#rimuovere alcuni specifici termini
corpus=tm_map(corpus, removeWords,c("dyson","hair", "dryer","'s", "even",
"is","the", "a", "hello"))
#stemming
corpus=tm_map(corpus, stemDocument)
```

Polaryzed tag plot

```
# one DTM by the two cleaned documents
tdm <- TermDocumentMatrix(corpus)
tdm.m <- as.matrix(tdm)
colnames(tdm.m) = c("Positive Comments", "Negative Comments")
# Cercare le parole comuni nei due documenti
common.words <- subset(tdm.m, tdm.m[, 1] > 0 & tdm.m[, 2] > 0)
difference <- abs(common.words[, 1] - common.words[, 2])
common.words <- cbind(common.words, difference)
```

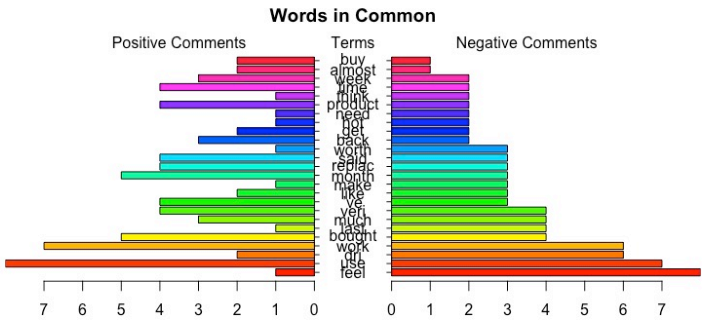
Polaryzed tag plot

Considering only the first 25 more frequent terms in both documents

```
top25.df <- data.frame(x = common.words[1:25, 2],y =common.words[1:25, 3],
labels =rownames(common.words[1:25, ]))
```

#polaryzed tag plot (or pyramid plot)

```
pyramid.plot(top25.df$x, top25.df$y,
labels = top25.df$labels, gap = 1, top.labels = c("Positive Comments",
                          "Terms", "Negative Comments"),
              main = "Words in Common", laxlab = NULL,
              raxlab = NULL, unit = NULL)
```



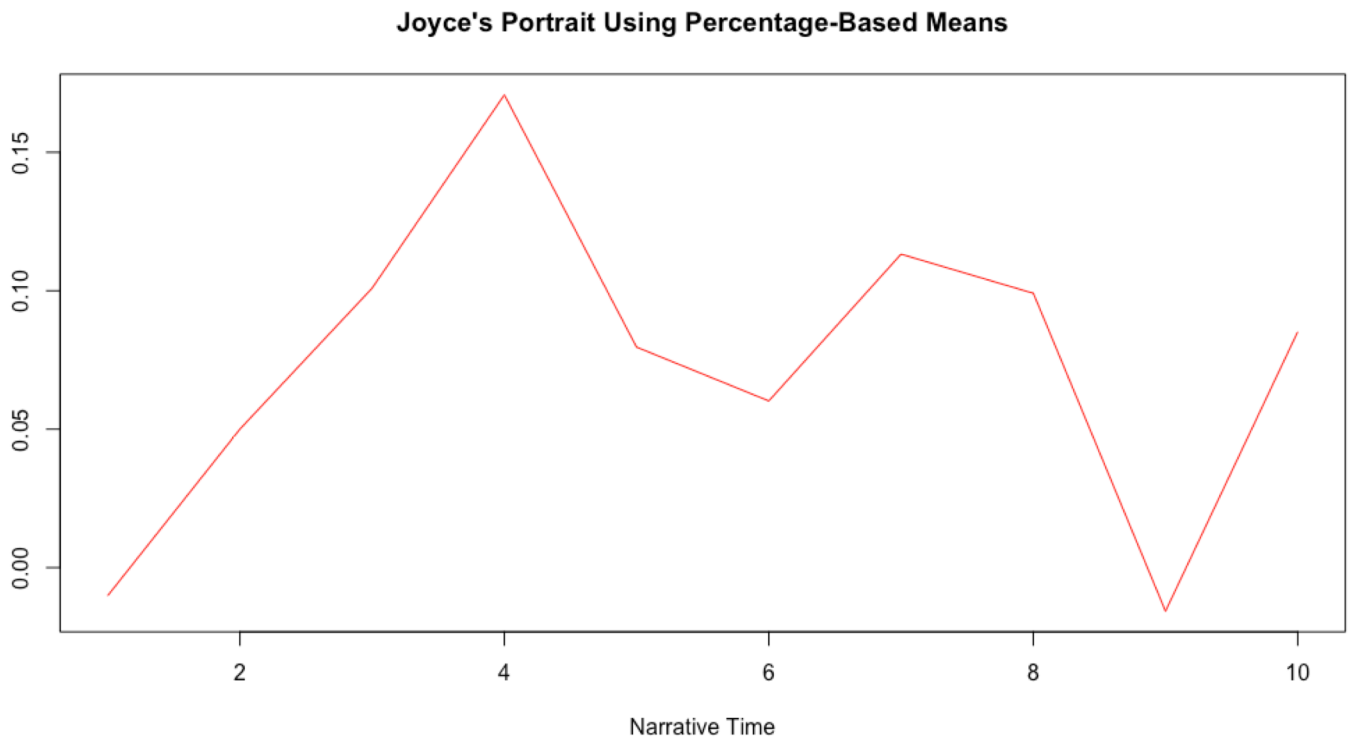
Sentiment Analysis in R

There are several methods of sentiment extraction from English language texts: Syuzhet, Bing, Affine e nrc.

```
# regular sentiment score using get_sentiment() function and method of
your choice. Comments is the original input file

syuzhet_vector <- get_sentiment(comments, method="syuzhet")

# see the first row of the vector

head(syuzhet_vector)

# see summary statistics of the vector

summary(syuzhet_vector)


# see the emotional variance associated to each sentence

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

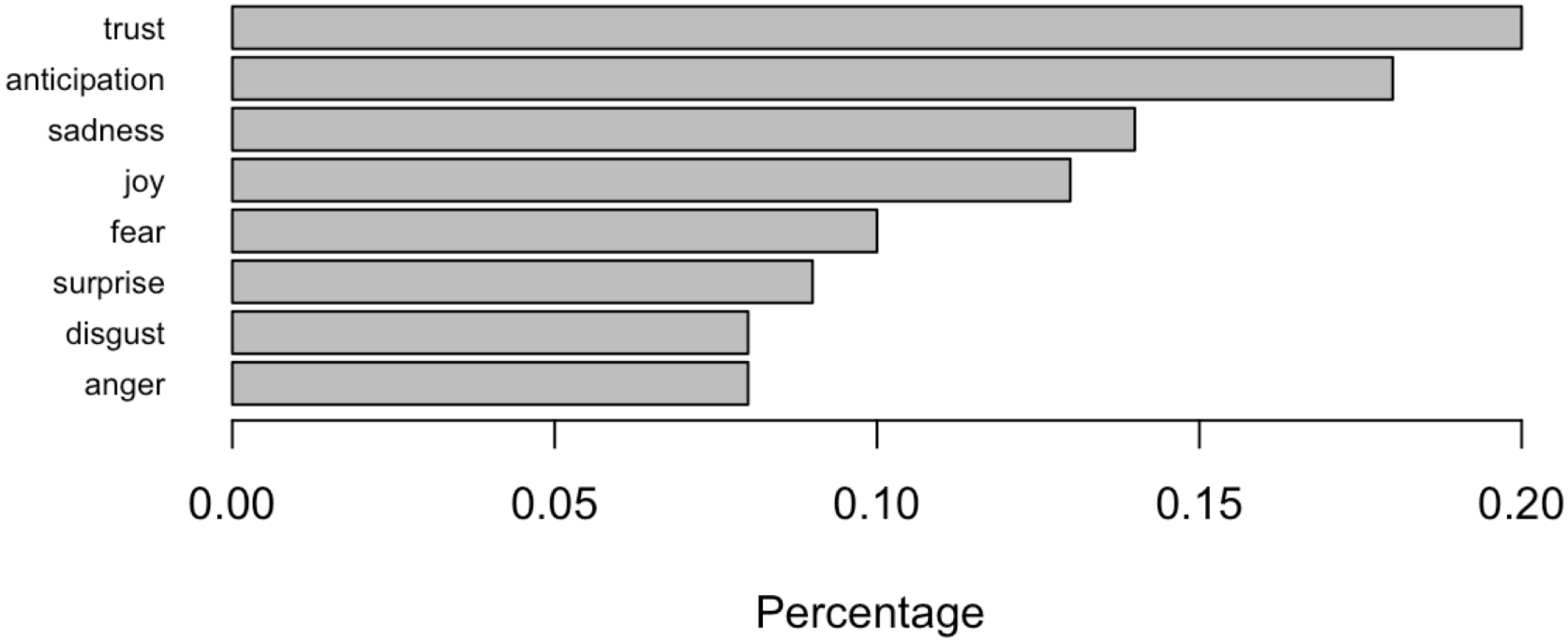
Sentiment Analysis in R

```
percent_vals <- get_percentage_values(syuzhet_vector, bins = 10)
plot(percent_vals, type="l", main="Joyce's Portrait Using Percentage-Based
Means", xlab = "Narrative Time", ylab= "Emotional Valence", col="red")
```



Sentiment Analysis in R

```
nrc_data <- get_nrc_sentiment(s_v)
barplot(sort(colSums(prop.table(nrc_data[, 1:8]))), horiz = TRUE, cex.names =
0.7, las = 1, main = "Emotions in Sample text", xlab="Percentage")
```



Sentiment scores

```
Sentimentscores=data.frame(col
ums(nrc_data))

names(Sentimentscores)<-
"Scores"

Sentimentscores<-
cbind("Sentiment"=row.names(Se
timentscores), Sentimentscores

rownames(Sentimentscores)<-NUL

ggplot(data = Sentimentscores,
aes(x=Sentiment,
y=Scores))+geom_bar(aes(fill=
Sentiment), stat = "identity")
theme(legend.position = "none"
+ xlab("Sentiments")+
ylab("Scores") +
ggtitle("Sentiments of people
behind the comments on d
company")
```