National College of Ireland

Configuration Manual

MSc Research Project
MSCDATOP

Liam Higgins
Student ID: x21182523

School of Computing
National College of Ireland

Supervisor: Jorge Basilio

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Liam Higgins
Student ID:	x21182523
Programme:	MSCDATOP **Year:** 2022
Module:	MSc Research Project
Supervisor:	Jorge Basilio
Submission Due Date:	15th August 2022
Project Title:	Analysing Airline Customer Experience using Lexicon and Machine Learning Sentiment Analyses of Twitter Data
Word Count:	815 **Page Count:** 11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Liam Higgins

Date: 13th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Liam Higgins
Student ID: x21182523

1 Introduction

This configuration manual describes the technical procedures to carry out the suggested research and produce the appropriate, reproducible findings. The handbook offers information on system configuration requirements, procedures for gathering and cleaning data, processes for creating lexicon and machine learning sentiment analysis models, and actions and code samples for model evaluation.

2 Hardware Configuration

The hardware specifications used for the implementation of the research project is detailed below.

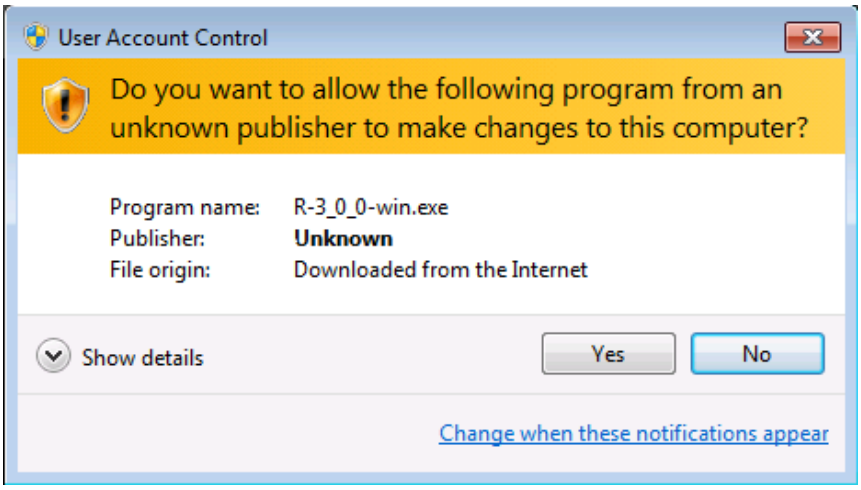
Operating System:	Microsoft Windows 10 Pro
Version:	10.0.19044 Build 19044
OS Manufacturer	Microsoft Corporation
System Name:	STULAP-4086
System Manufacturer:	Dell Inc.
System Model:	Latitude 5410
System Type:	x64-based PC
System SKU:	09A0
Processor:	Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz, 2112 Mhz, 4 Core(s), 8 Logical Processor(s)
BIOS Version/Date:	Dell Inc. 1.10.0, 14/12/2021
SMBIOS Version:	3.2
BIOS Mode:	UEFI
Windows Directory:	C:\Windows
System Directory:	C:\Windows\system32
Installed Physical Memory:	16.0 GB
Total Physical Memory	15.6 GB
Available Physical Memory	5.29 GB
Total Virtual Memory:	18.0 GB
Available Virtual Memory:	6.59 GB
Page File Space:	2.38 GB
Page File:	C:\pagefile.sys
Kernel DMA Protection	On
Hyper-V enabled:	Yes

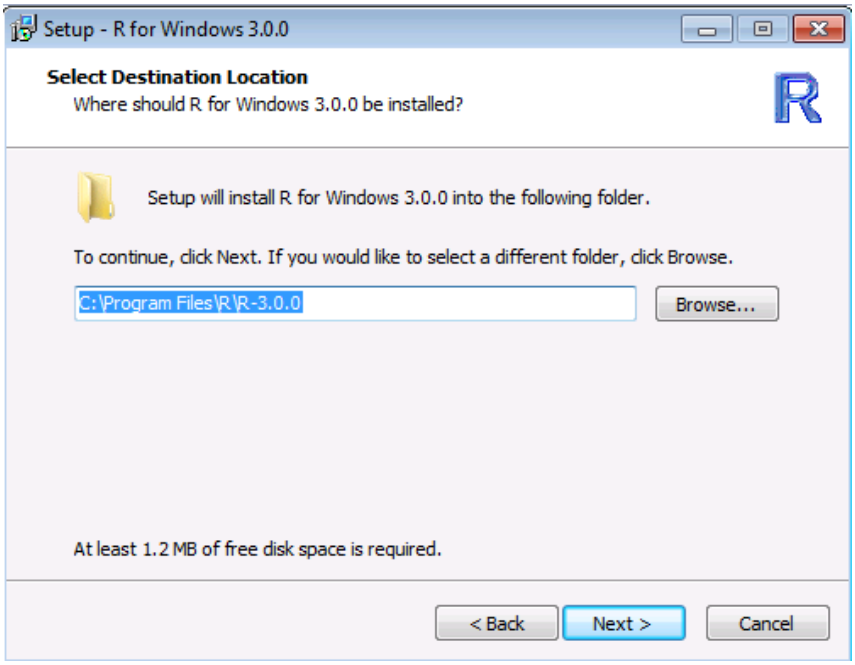
3 Software Configuration

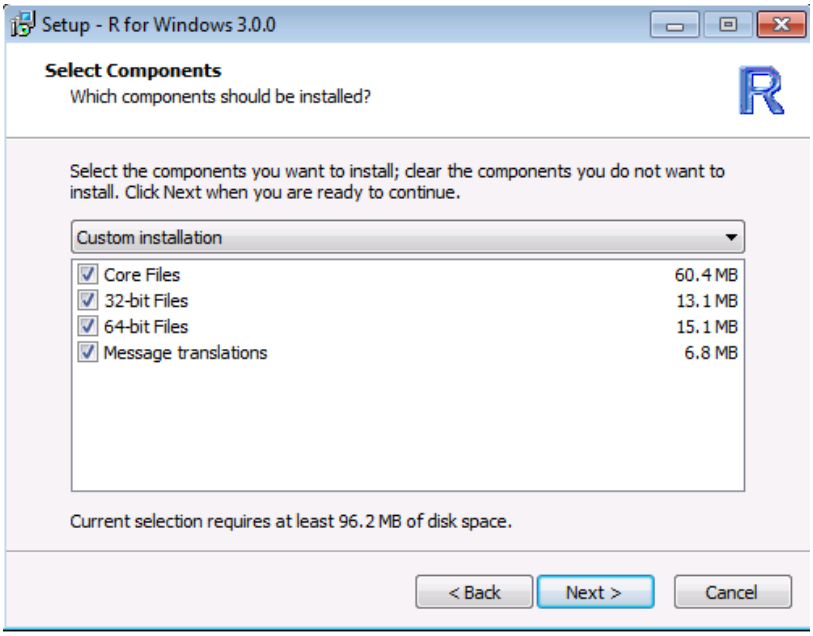
All coding was conducting using the R programming language. The integrated development environment (IDE) for R is called RStudio. Along with a console-based editor with syntax highlighting and direct code execution, it also has features for managing workspaces and graphing, history, debugging, and history. RStudio runs on desktop computers and is offered in both open source and paid editions (Windows, Mac, and Linux). The open source free to use version was used.

3.1 Installing R Language

The R language is available via CRAN file transfer protocol (FTP) servers (https://cran.r-project.org/) and below is the method to download and install on a Windows 10 laptop.

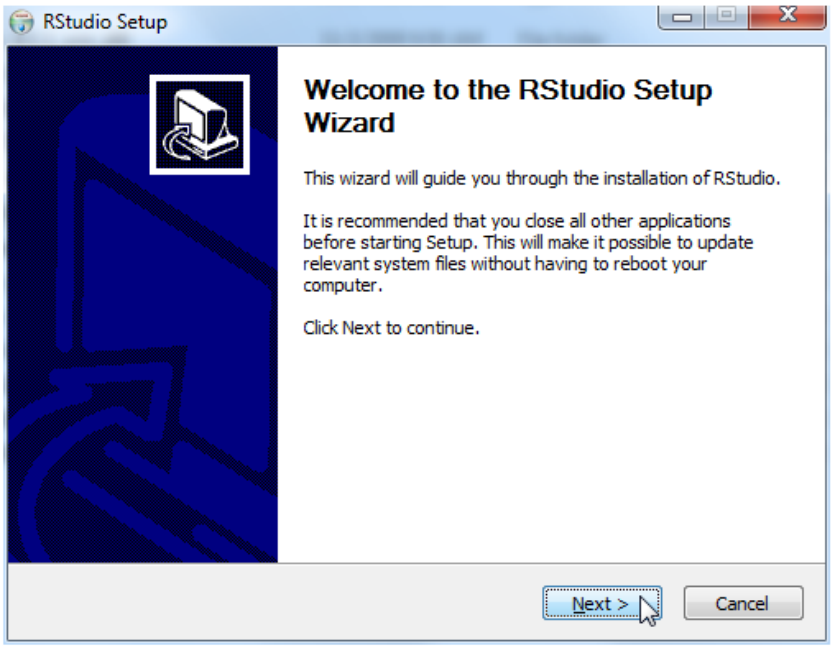


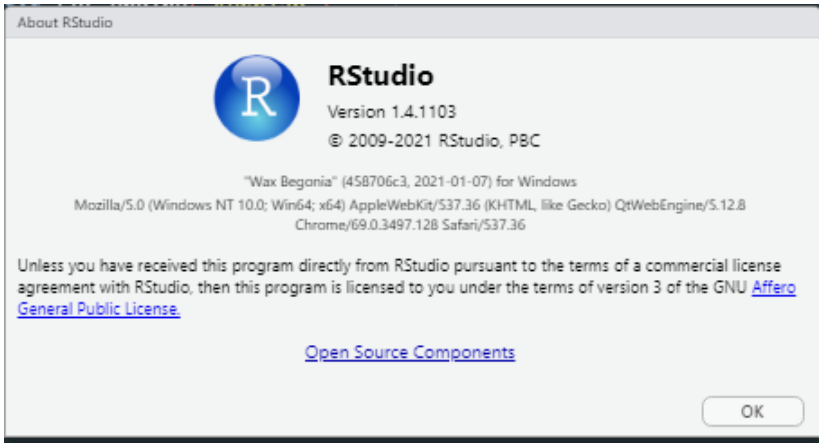




3.2 RStudio Download & Installation

RStudio is an open source Integrated Development Environment (IDE) freely available at the official RStudio website (https://www.rstudio.com/products/rstudio/download/). By selecting the appropriate Windows 10 compatible version download it is necessary to run the installer.





3.3 Installation of Required Libraries & Packages

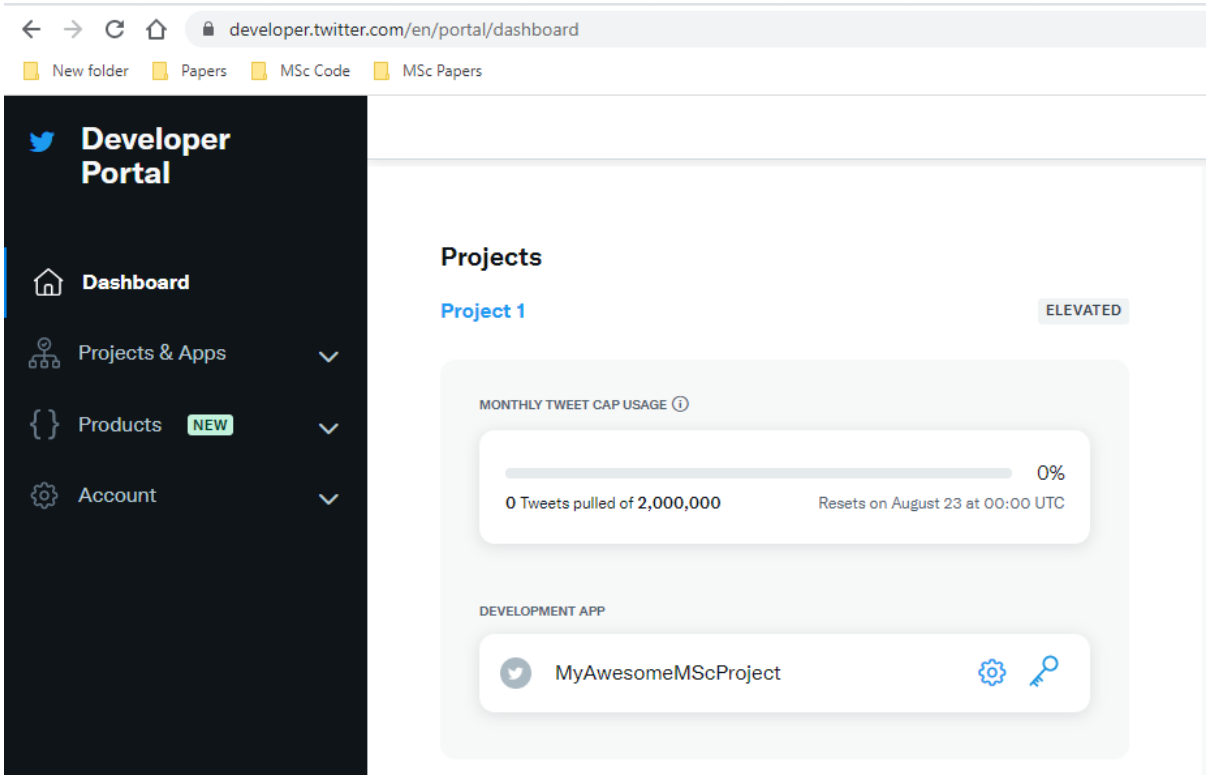
R Library:	Library Function:	Library Documentation:
library(readr)	Reading & writing of csv files	https://www.rdocumentation.org/packages/readr/versions/1.3.1
library(tidyr)	Reshape & manipulation	https://www.rdocumentation.org/packages/tidyr/versions/0.8.3
library(dplyr)	Data Manipulation	https://www.rdocumentation.org/packages/dplyr/versions/0.7.8
library(ggpubr)	Visualisations	https://www.rdocumentation.org/packages/ggpubr/versions/0.4.0
library(GGally)	Plots & Graphs	https://www.rdocumentation.org/packages/GGally/versions/1.5.0
library(mlbench)	Machine Learning Benchmark Problems	https://www.rdocumentation.org/packages/mlbench/versions/2.1
library(caret)	Classification And Regression Models	https://www.rdocumentation.org/packages/caret/versions/6.0
library(heatmaply)	Visualizing high-dimensional data	https://www.rdocumentation.org/packages/heatmaply/versions/1.2.1
library(tidyverse)	Tidy Data Outputs	https://www.rdocumentation.org/packages/tidyverse/versions/1.3.0
library(twitteR)	Connecting to Twitter API	https://cran.r-project.org/web/packages/twitteR/twitteR.pdf
library(rtweet)	Collecting Twitter Data	https://cran.r-project.org/web/packages/rtweet/rtweet.pdf
library(ROAuth)	Interface For OAuth	https://cran.r-project.org/web/packages/ROAuth/index.html
library(hms)	Formatting date and time-of-day	https://cran.r-project.org/web/packages/hms/index.html
library(lubridate)	Date formatting	https://cran.r-project.org/web/packages/lubridate/vignettes/lubridate.html
library(tidytext)	Text mining and tidy data frames	https://cran.r-project.org/web/packages/tidytext/index.html
library(tm)	Text Mining	https://www.rdocumentation.org/packages/tm/versions/0.7-8
library(wordcloud)	Word Cloud Generator	https://cran.r-project.org/web/packages/wordcloud/wordcloud.pdf
library(igraph)	Network Analysis and Visualization	https://igraph.org/r/pdf/latest/igraph.pdf
library(glue)	Interpreting String Literals	https://cran.r-project.org/web/packages/glue/index.html
library(stringr)	String manipulation	https://www.rdocumentation.org/packages/stringr/versions/1.4.0
library(ggplot2)	Plots	https://www.rdocumentation.org/packages/ggplot2/versions/3.3.3
library(plotly)	Plots	https://plotly.com/python/
library(ggeasy)	Easy Access to 'ggplot2' Commands	https://cran.r-project.org/web/packages/ggeasy/index.html

library(SnowballC)	Text Stemming	https://cran.r-project.org/web/packages/SnowballC/SnowballC.pdf
library(RColorBrewer)	Colour Palettes	https://www.rdocumentation.org/packages/RColorBrewer/versions/1.1-2
library(magrittr)	Better structuring of data	https://www.rdocumentation.org/packages/magrittr/versions/2.0.1
library(Amelia)	Imputes missing data	https://www.rdocumentation.org/packages/Amelia/versions/1.7.6/topics/missmap
library(forcats)	GGplot frequency	https://cran.r-project.org/web/packages/forcats/index.html
library(caTools)	Data wrangling	https://cran.r-project.org/web/packages/caTools/caTools.pdf
library(gridExtra)	Arranging plots into grids & labeling	https://www.rdocumentation.org/packages/gridExtra/versions/2.3
library(cowplot)	Arranging plots into grids & labeling	https://www.rdocumentation.org/packages/cowplot/versions/1.1.1
library(reshape2)	Transform wide/long data	https://www.rdocumentation.org/packages/reshape2/versions/1.4.4
library(scales)	Better plotting	https://www.rdocumentation.org/packages/scales/versions/0.4.1
library(rpart)	CART algorithms	https://cran.r-project.org/web/packages/rpart/rpart.pdf
library(rpart.plot)	Plotting Decision Trees	https://cran.r-project.org/web/packages/rpart.plot/index.html
library(partykit)	Visualising Decision Trees	https://cran.r-project.org/web/packages/partykit/index.html
library(data.table)	Subset rows, select and compute columns	https://www.rdocumentation.org/packages/data.table/versions/1.13.6
library(ggthemes)	Themes for visuals	https://www.rdocumentation.org/packages/ggthemes/versions/3.5.0
library(mice)	Imputation & other tools	https://www.rdocumentation.org/packages/mice/versions/3.13.0
library(widyr)	Matrix Functions	https://cran.r-project.org/web/packages/widyr/index.html
library(lexicon)	lexicons, dictionaries, and word lists	https://cran.r-project.org/web/packages/lexicon/index.html
library(sentimentr)	Calculate Text Polarity Sentiment	https://cran.r-project.org/web/packages/sentimentr/sentimentr.pdf
library(syuzhet)	Extracts sentiment syuzhet lexicon	https://cran.r-project.org/web/packages/syuzhet/index.html
library(stopwords)	Remove stopwords for NLP	https://cran.r-project.org/web/packages/stopwords/index.html
library(e1071)	Machine Learning algorithms	https://cran.r-project.org/web/packages/e1071/index.html
library(createDataPartition)	Split data (test/train)	https://www.rdocumentation.org/packages/caret/versions/6.0-92/topics/createDataPartition
library(rsample)	Bootstrap and cross-validation	https://cran.r-project.org/web/packages/rsample/index.html
library(recipes)	Prepares data for modeling	https://cran.r-project.org/web/packages/recipes/index.html
library(textrecipes)	Converts text to numerical features	https://cran.r-project.org/web/packages/textrecipes/index.html
library(yardstick)	Confusion matrix and RMSE functions	https://cran.r-project.org/web/packages/yardstick/index.html
library(kernlab)	Kernel-Based Machine Learning Algorythms	https://cran.r-project.org/web/packages/kernlab/index.html

3.4 Twitter Developer Account

To access the Twitter Application Programming Interface (API) and extract the Twitter Tweet data needed for this research project a Twitter Developer Account is required. This can be obtained free of charge by visiting https://developer.twitter.com/en and registering using a valid email address.

A “Project” is required once a developer account has been accepted. Work can be divided into projects based on how you plan to use the Twitter API, making it easier to manage your access to the API and keep track of your usage. Each Project has an App that allows you to create the login information needed to use the Twitter API. In our documentation's getting started section, you can find out more about how to begin using the Twitter API.



3.5 Connecting to the Twitter API from RStudio

```
# Connect to Twitter API App using Developer Account
my_authorization <- rtweet::create_token(app = "MyAwesomeMScProject",
                                         consumer_key = "z0QjY3s4j5MMWky27s1iwo394",
                                         consumer_secret = "Y1BckhuUMDPUPpySJKqIOEmqqWN8O78poJIivMsuOhobHleCws",
                                         access_token="1506717716790534149-3b2CPmKauiKq2ithVcsnWfaIpQVuv9",
                                         access_secret = "EVzWSpvDojLfWktYhlDvgy5AT5XLw1ZP5cKQBsR7ZzkF0")
```

3.6 Extracting Twitter Data

```
# Extract Select Airline Tweets
tweets <- rtweet::get_timeline(c("ryanair", "aerlingus", "delta", "unitedairlines", "britishairways",
                                 "emirates", "lufthansa"), n = 1000, parse=T, token=my_authorization)
rtweet::write_as_csv(tweets, "air_tweets2.csv", prepend_ids = TRUE, na = "", fileEncoding = "UTF-8")
```

3.7 Exploring the Data

```
#################################################################################################
# EXPLORE THE DATASET STRUCTURE & VARIABLES
#################################################################################################
# Check class of imported airline_tweets data & whether any metadata
attributes(data)
# Check head of df
head(data)
# Check dimensions of df
dim(data)
# Check structure of df
str(data)
# Glimpse overall df
glimpse(data)
# Get a summary of which type of variables are contained in each column & some basic statistics
summary(data)
# Check number of negative, neutral and positive sentiments
# table(data$Category)
# Selection of Limited Columns:
reducedData <- select(data, user_id,created_at,screen_name, text)
# Double check column names
names(reducedData)
# Quick view of DF
View(reducedData)
```

3.8 Creating Text Corpus & Cleaning the Data

```
#################################################################################################
# CREATING A TEXT CORPUS & CLEANING THE DATA
#################################################################################################
# Load the data as a corpus
TextDoc <- Corpus(VectorSource(reducedData$text))

# Data Cleaning
#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, "\\|")
# Convert the text to lower case
TextDoc <- tm_map(TextDoc, content_transformer(tolower))
# Remove numbers
TextDoc <- tm_map(TextDoc, removeNumbers)
# Remove common english stopwords
TextDoc <- tm_map(TextDoc, removeWords, stopwords("english"))
# Remove custom own stop words
TextDoc <- tm_map(TextDoc, removeWords, c("s", "company", "team", "tco", "https", "sorri", "plea"))
# Remove punctuations
TextDoc <- tm_map(TextDoc, removePunctuation)
# Eliminate extra white spaces
TextDoc <- tm_map(TextDoc, stripWhitespace)
# Text stemming - which reduces words to their root form
TextDoc <- tm_map(TextDoc, stemDocument)
```

3.9 Creating Term Document Matrix

```
#################################################################################################
# BUILDING A TERM DOCUMENT MATRIX
#################################################################################################
# Build a term-document matrix
TextDoc_dtm <- TermDocumentMatrix(TextDoc)
TextDoc_dtm
# Convert to Matrix
dtm_m <- as.matrix(TextDoc_dtm)
# Check first ten words and 1 to 20 columns.
dtm_m[1:10, 1:20]

# Find out how often each word appears
w <- rowSums(dtm_m)
# Create subset of word frequencies great or equal to 25
w <- subset(w, w>=100)
w
barplot(w, las = 2, col = rainbow(20))

# Sort by descearing 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)
```

3.10 Initial Barplot & World Cloud

```
#################################################################################################
# PLOTS & WORDCLOUD
#################################################################################################
# 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")

#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"))
```

3.11 Initial Barplot & World Cloud

```
#################################################################################################
# CREATING A REDUCED SET OF DATA FOR SENTIMENT ANALYSIS
#################################################################################################
# Clean reducedData
removeUsername <- gsub('@[^[:space:]]*', '', reducedData$text)
reducedData$text <- gsub("/", " ", reducedData$text)
reducedData$text <- gsub("@", " ", reducedData$text)
reducedData$text <- gsub("\\|", " ", reducedData$text)
reducedData$text <- tolower(reducedData$text)
```

3.12 Sentiment Analysis using Syuzhet, Bing & AFINN Lexicons

```
###################################################################################################
# SENTIMENT ANALYSIS WITH 3 LEXICON DICTIONARIES: SYUZHET, BING & AFINN
###################################################################################################
# Sentiment Score
# regular sentiment score using get_sentiment() function and method of your choice
# please note that different methods may have different scales
syuzhet_vector <- get_sentiment(reducedData$text, method="syuzhet")
# see the first row of the vector
head(syuzhet_vector)
# see summary statistics of the vector
summary(syuzhet_vector)

# bing
bing_vector <- get_sentiment(reducedData$text, method="bing")
head(bing_vector)
summary(bing_vector)

# afinn
afinn_vector <- get_sentiment(reducedData$text, method="afinn")
head(afinn_vector)
summary(afinn_vector)

# compare the first row of each vector using sign function
rbind(
  sign(head(syuzhet_vector)),
  sign(head(bing_vector)),
  sign(head(afinn_vector))
)
```

3.13 Sentiment Analysis using NRC Lexicon

```
###################################################################################################
# SENTIMENT ANALYSIS WITH NRC Emotion Lexicon
###################################################################################################
# NRC sentiment analysis to return data frame with each row classified as one of the following
# emotions, rather than a score: anger, anticipation, disgust, fear, joy, sadness, surprise, trust
# It also counts the number of positive and negative emotions found in each row
# WARNING - TAKES SEVERAL MINUTES
d<-get_nrc_sentiment(reducedData$text)
# head(d,10) - to see top 10 lines of the get_nrc_sentiment dataframe
head (d,10)
# transpose to dataframe
td<-data.frame(t(d))
#The function rowSums computes column sums across rows for each level of a grouping variable.
td_new <- data.frame(rowSums(td[2:253]))
#Transformation and cleaning
names(td_new)[1] <- "count"
td_new <- cbind("sentiment" = rownames(td_new), td_new)
rownames(td_new) <- NULL
td_new2<-td_new[1:8,]
# Adding sentiment to the dataframe
reducedData_sentiment <- reducedData
reducedData_sentiment$sentiment_afinn_numeric <- afinn_vector
reducedData_sentiment$sentiment_bing_numeric <- bing_vector
reducedData_sentiment$sentiment_syuzhet_numeric <- syuzhet_vector
# Add labels for positive/negative/neutral
# afinn
reducedData_sentiment$sentiment_afinn_label <- ifelse(reducedData_sentiment$sentiment_afinn_numeric > 0, "positive", ifelse(
# bing
reducedData_sentiment$sentiment_bing_label <- ifelse(reducedData_sentiment$sentiment_bing_numeric > 0, "positive", ifelse(re
# syuzhet
reducedData_sentiment$sentiment_syuzhet_label <- ifelse(reducedData_sentiment$sentiment_syuzhet_numeric > 0, "positive", ife
# Check dataframe
head(reducedData_sentiment)
# Look at similarity between methods
reducedData_sentiment$matching_sentiment_result <- reducedData_sentiment$sentiment_afinn_label == reducedData_sentiment$sent
# Table of TRUE & FALSE between methods
# TRUE indicates that all the methods give the same result, FALSE means they give different results
prop.table(table(reducedData_sentiment$matching_sentiment_result))
```

3.14 Plotting the Sentiment Analyses Results

```
###################################################################################
# PLOTS FOR SENTIMENTS
###################################################################################
# sentiment_afinn_label plot
negative_tweets <- as.data.frame(prop.table(table(reducedData_sentiment[,c("screen_name","sentiment_afinn_label")]))*100)
colnames(negative_tweets) <- c("Airline","Airline_Sentiment","Proportion_of_Sentiment")
ggplot(negative_tweets, aes(x = Airline, y = Proportion_of_Sentiment,fill = Airline_Sentiment)) +
  geom_bar(stat="identity", colour = "white", position = "dodge") +
  scale_fill_brewer(palette = "Set1") +
  geom_text(aes(label=round(Proportion_of_Sentiment,digits=2), vjust=-0.5),
            position = position_dodge(width = 1))

# sentiment_bing_label plot
negative_tweets <- as.data.frame(prop.table(table(reducedData_sentiment[,c("screen_name","sentiment_bing_label")]))*100)
colnames(negative_tweets) <- c("Airline","Airline_Sentiment","Proportion_of_Sentiment")
ggplot(negative_tweets, aes(x = Airline, y = Proportion_of_Sentiment,fill = Airline_Sentiment)) +
  geom_bar(stat="identity", colour = "white", position = "dodge") +
  scale_fill_brewer(palette = "Set1") +
  geom_text(aes(label=round(Proportion_of_Sentiment,digits=2), vjust=-0.5),
            position = position_dodge(width = 1))

# sentiment_syuzhet_label plot
negative_tweets <- as.data.frame(prop.table(table(reducedData_sentiment[,c("screen_name","sentiment_syuzhet_label")]))*100)
colnames(negative_tweets) <- c("Airline","Airline_Sentiment","Proportion_of_Sentiment")
ggplot(negative_tweets, aes(x = Airline, y = Proportion_of_Sentiment,fill = Airline_Sentiment)) +
  geom_bar(stat="identity", colour = "white", position = "dodge") +
  scale_fill_brewer(palette = "Set1") +
  geom_text(aes(label=round(Proportion_of_Sentiment,digits=2), vjust=-0.5),
            position = position_dodge(width = 1))

#Plot the count of words associated with each sentiment
quickplot(sentiment, data=td_new2, weight=count, geom="bar", fill=sentiment, ylab="count")+ggtitle("Tweet Emotions")

#Plot the count of words associated with each sentiment, expressed as a percentage
barplot(sort(colSums(prop.table(d[, 1:8]))), horiz = TRUE, cex.names = 0.7, col = rainbow(50),las = 1,
        main = "Emotions in Text", xlab="Percentage")
```

3.15 Creating Test & Train Subsets

```
###################################################################################################
# TEST & TRAIN SETS
###################################################################################################
# Split data into test & train sets
set.seed(99)
split = sample.split(model_data$sentiment,SplitRatio = 0.7)
train = subset(model_data,split = TRUE)
test = subset(model_data,split = FALSE)
glimpse(train)
glimpse(test)
str(train)
str(test)
# Convert y variable "sentiment" to a factor in the test & train sets
train$sentiment <- as.character(train$sentiment)
train$sentiment <- as.factor(train$sentiment)
test$sentiment <- as.character(test$sentiment)
test$sentiment <- as.factor(test$sentiment)
```

3.16 Decision Tree Model

```
#################################################################################################
# DECISION TREE
#################################################################################################
library(rpart)
library(rpart.plot)
# Build the first Decision Tree model
my_tree <- rpart(sentiment ~., method = 'class', data = train)
# Plot the Decision Tree model
rpart.plot(my_tree, extra = 106)
# Build prediction tree
predict_tree <-predict(my_tree, data = test, type = 'class')
# Create cross table with results of prediction
tree_table <- table(test$sentiment, predict_tree)
tree_table
# Calculate accuracy from diagonal values from cross table
accuracy_Test <- sum(diag(tree_table)) / sum(tree_table)
print(paste('Accuracy for test', accuracy_Test)) # "Accuracy for test 0.737590072057646"
# Confusion Matrix
confusionMatrix(test$sentiment, predict_tree) # Accuracy : 0.7376
# Set control parameters for tree tuning
control <- rpart.control(minsplit = 4, minbucket = round(5 / 3), maxdepth = 3, cp = 0)
# Build tuned tree
tune_fit <- rpart(sentiment~., data = train, method = 'class', control = control)
# Plot tree
rpart.plot(tune_fit, extra = 106)
# Create Predictions
predict_tree2 <-predict(tune_fit, data = test, type = 'class')
# Output in cross table
tree_table2 <- table(test$sentiment, predict_tree2)
tree_table2
# Calculate accuracy from diagonal sums of cross table output
accuracy_Test2 <- sum(diag(tree_table2)) / sum(tree_table2)
# Print accuracy
print(paste('Accuracy for test2', accuracy_Test2)) # "Accuracy for test2 0.609887910328263"
# Confusion Matrix with full model outputs
confusionMatrix(test$sentiment, predict_tree2) # Accuracy : 0.6099
```

3.17 Naïve Bayes Model

```
#################################################################################################
# NAIVE BAYES
#################################################################################################
# Convert y variable "sentiment" to a factor in the test & train sets
# training$Category <- as.character(training$Category)
training$Category <- as.factor(training$Category)
# testing$Category <- as.character(testing$Category)
testing$Category <- as.factor(testing$Category)

# Set variables for Navie Bayes model
x <- training[,-3]
y<- training$Category
# Build NB model
nb = train(x,y,'nb',trControl=trainControl(method='cv',number=10))
nb
Predict <- predict(nb,newdata = testing )
confusionMatrix(Predict,testing$Category )

# Create predictions on testing data
Predict <- predict(nb,newdata = testing)
# Evaluate results via Confusion matrix
confusionMatrix(Predict,testing$Category, positive = "Positive")

# Plot area under the curve
library(pROC)
ROCurve<-roc(testing$Category,as.numeric(Predict))
# Area Under the Curve
auc(ROCurve)
## Plot the Receiver Operating Characteristic
plot(ROCurve)
```