#install.packages('RCurl', dep = TRUE)

#install.packages('plyr', dep = TRUE)

#install.packages('XML', dep = TRUE)

#install.packages('stringr', dep = TRUE)

#install.packages('dplyr', dep = TRUE)

#install.packages('knitr', dep = TRUE)

#install.packages('purrr', dep = TRUE)

#install.packages('stringr', dep = TRUE)

#install.packages('rvest', dep = TRUE)

#install.packages('magrittr', dep = TRUE)

#install.packages('tidytext', dep = TRUE)

#install.packages('dplyr', dep = TRUE)

#install.packages('igraph', dep = TRUE)

#install.packages('ggraph', dep = TRUE)

#install.packages('imager', dep = TRUE)

#install.packages('RColorBrewer', dep = TRUE)

#install.packages('highcharter', dep = TRUE)

# install.packages('gtable', dep = TRUE)

# install.packages('jpeg', dep = TRUE)

# install.packages('imager', dep = TRUE)

library(RCurl)

library(plyr)

library(XML)

library(stringr)

library(dplyr)

library(knitr)

library(purrr)

library(stringr)

library(rvest)

library(magrittr)

library(tidytext)

library(dplyr)

library(gtable)

library(igraph)

library(ggraph)

library(imager)

library(RColorBrewer)

library(highcharter)

#Set my working directory

setwd("~/Saint/Regis/Data Science Practicum II - MSDS 696 X 70/Assignment")

# Retrieve Billboard charted songs from wikipedia

rihannaSongs <- data.frame()

for (i in 2005:2016)

{

# Different URLs for every yr

URL <- paste("http://en.wikipedia.org/wiki/Billboard\_Year-End\_Hot\_100\_singles\_of\_",i,sep="")

# HTML shredding / parsing

results <- htmlTreeParse(getURL(URL, followlocation=TRUE), useInternal=TRUE)

billboardChartData <- xpathSApply(results, "//table[@class='wikitable sortable']//tr",xmlValue)

split\_billboardChartData <- str\_split\_fixed(billboardChartData,"\n",3)

billboard <- as.data.frame(cbind(split\_billboardChartData[2:101, ], rep(i,100)), stringsAsFactors=FALSE)

# row bind this year's data to all the data

rihannaSongs <- rbind(rihannaSongs, billboard)

}

colnames(rihannaSongs) <- c("Rank", "Song", "Artist", "Year")

rihannaSongs$Song <- gsub('\\"', "", rihannaSongs$Song)

rihannaSongs$Song <- tolower(gsub("[^[:alnum:] ]", "", rihannaSongs$Song))

rihannaSongs$Song <- gsub("\\'", "", iconv(rihannaSongs$Song, to='ASCII//TRANSLIT')) # accent character correction

rihannaSongs$Artist <- tolower(gsub("[^[:alnum:] ]", "", rihannaSongs$Artist))

rihannaSongs$Artist <- gsub("'e", "e", iconv(rihannaSongs$Artist, to='ASCII//TRANSLIT')) # accent character correction

rihannaSongs$Artist<- gsub("'o", "o", rihannaSongs$Artist)

# Keep only songs done by Riri. The processing kills my laptop, if I try to do either all artists, Defjam, male, or female artists.

rihannaSongs <- filter(rihannaSongs, Artist == "rihanna")

# Lyrics and Source variables. Source variable will be used in my next analysis, which will be after this course ends.

rihannaSongs$Lyrics <- ""

rihannaSongs$Source <- ""

for (a in 1:length(rihannaSongs$Song))

{

lyrics <- "RiRi lyrics will soon be assigned"

results <- 313 # Use any number below

# For URL use only main artist

artist <- strsplit(rihannaSongs$Artist[a], " and | with | featuring | duet | feat. | with | feat ")

artist <- unlist(artist)[[1]]

onlyRiRi <- gsub("the ", "", artist)

leftString1 <- substring(onlyRiRi, 1, 1)

# Build the URLs

metroURL <- paste("http://metrolyrics.com/",rihannaSongs$Song[a],"-lyrics-",onlyRiRi,".html",sep="")

songURL <- paste("http://songlyrics.com/",onlyRiRi,"/",rihannaSongs$Song[a],"-lyrics",sep="")

modeURL <- paste("http://www.lyricsmode.com/lyrics/", leftString1, "/", onlyRiRi, "/", rihannaSongs$Song[a], ".html", sep="")

URLs <- c(metroURL, songURL, modeURL)

lyriclocs <- c("//div[@id='lyrics-body-text']", "//p[@id='songLyricsDiv']", "//p[@id='lyrics\_text']")

for (k in 1:length(URLs))

{

rihannaSongs$Lyrics[a] <- "Not set yet."

results <- 13 # Use any number

if(k!=3) URL <- tolower(gsub(" ", "-", URLs[k]))

if(k==3) URL <- URLs[k]

tryCatch({

results <- htmlTreeParse(URL, useInternal=TRUE, isURL=TRUE)

lyrics <- xpathSApply(results, lyriclocs[k], xmlValue) },

error = function(x) {

message(paste(a, "Missing")) },

finally={

if (!is.numeric(results)) {

if (length(lyrics)!=0) {

rihannaSongs$Lyrics[a] <- lyrics[[1]]

message(paste(a, "Found"))

rihannaSongs$Source[a] <- k

break

}

}

}) # tryCatch ends

} # url FOR(k) ends

} # lyrics FOR(a) ends

rihannaSongs$Lyrics <- gsub("\\\n|\\\t"," ",rihannaSongs$Lyrics)

rihannaSongs$Lyrics <- tolower(gsub("[^[:alnum:] ]", "", rihannaSongs$Lyrics))

missing <- round(length(rihannaSongs[rihannaSongs$Lyrics=="Will be assigned", 1])/length(rihannaSongs[,1]), 4)\*100

## Add lyrics to the dataframe

rihannaSongs$Lyrics <- gsub("To be assigned", "NA", rihannaSongs$Lyrics)

rihannaSongs$Lyrics <- gsub("Lyrics unavailable because of license restrictions", "NA", rihannaSongs$Lyrics)

#Store the songs

write.csv(rihannaSongs, "rihannaLyrics2005To2015.csv", row.names=FALSE)

rihannaSongs <- read.csv("rihannaLyrics2005To2015.csv", stringsAsFactors=FALSE)

# Remove source of lyrics and artist, then add row numbers

rihannaSongs <- subset(rihannaSongs, select=-Source)

rihannaSongs <- subset(rihannaSongs, select=-Artist)

rihannaSongs <- arrange(rihannaSongs, desc(Rank))

rihannaSongs$line <- sprintf("%3d", 1:nrow(rihannaSongs)) #add row numbers

#Remove “only girl in the world”, “diamonds”, “disturbia”, and “”, which are each on the chart twice. Lower Rank numbers kept.

rihannaSongs <- subset(rihannaSongs, Rank != 47 & Rank != 94 & Year != 2009)

# The get\_artist\_audio\_features() on Spotify was giving me problems with authentication.

#Therefore, I quickly added the album names to the dataframe, with copy-paste (not pretty) code.

rihannaSongs <-

mutate(rihannaSongs, Album =

ifelse(Song == "cheers drink to that", "Loud",

ifelse(Song == "diamonds", "Unapologetic",

ifelse(Song == "dont stop the music", "Good Girl Gone Bad",

ifelse(Song == "only girl in the world", "Loud",

ifelse(Song == "shut up and drive", "Good Girl Gone Bad",

ifelse(Song == "sm", "Loud",

ifelse(Song == "bitch better have my money", "No Album",

ifelse(Song == "disturbia", "Good Girl Gone Bad",

ifelse(Song == "only girl in the world", "Loud",

ifelse(Song == "pon de replay", "Music Of The Sun",

ifelse(Song == "pour it up", "Unapologetic",

ifelse(Song == "rude boy", "Rated R",

ifelse(Song == "sos", "A Girl Like Me",

ifelse(Song == "take a bow", "Good Girl Gone Bad",

ifelse(Song == "unfaithful", "A Girl like Me",

ifelse(Song == "where have you been", "Talk That Talk",

ifelse(Song == "you da one", "Talk That Talk",

ifelse(Song == "needed me", "Anti",

NA)))))))))))))))))))

#Count the most used words in the songs

rihannaSongWords <- unnest\_tokens(rihannaSongs, word, Lyrics, drop=TRUE) %>% anti\_join(stop\_words)

count(rihannaSongWords, word, sort = TRUE) %>% slice(1:3635) %>% kable(format = "markdown", row.names=FALSE) #Print word usage

#Get the most used pair of words in the songs

rihannaSongWordPairs <- unnest\_tokens(rihannaSongs, wordPairs, Lyrics, token = "ngrams", n = 2, drop= TRUE)

#Print and store the most used pair of words in the songs

count(rihannaSongWordPairs, wordPairs, sort = TRUE) %>% slice(1:3635) %>% kable(format = "markdown", row.names=FALSE)

ririWordPairCount <- count(rihannaSongWordPairs, wordPairs, Song, sort = TRUE)

#Create word pair usage graph

set.seed(1171413) # Any int works here

ririWordPairCount %>%

filter(n >= 30) %>%

graph\_from\_data\_frame() %>%

ggraph(layout = "fr") +

geom\_edge\_link(aes(edge\_alpha = n, edge\_width = n)) +

geom\_node\_point(color = "#0052A5", size = 5) +

geom\_node\_text(aes(label = name), vjust = 1.5) +

ggtitle(expression(paste("Popular word-pairs in Billboard year-end top 100 songs by ", italic("Rihanna")))) + theme\_void()

#Calculate anger per song by using the NRC (National Research Council Canada).

angryWords <- sentiments %>% filter(lexicon == 'nrc', sentiment == 'anger') %>% select(word) %>% mutate(anger = T)

ririAngryWordsInSongs <- rihannaSongWords %>% anti\_join(stop\_words, by = 'word') %>% left\_join(angryWords, by = 'word') %>% group\_by(Song) %>% summarise(AngerPct = round(sum(anger, na.rm = T) / n(), 4), CountOfWords = n()) %>% ungroup

ririAngryWordsInSongs %>% select(AngerPct, Song) %>% arrange(-AngerPct) %>% head(10)

ririSongAnalysis <- merge(rihannaSongWords , ririAngryWordsInSongs, by = "Song")

ririSongAnalysis <- ririSongAnalysis %>% distinct # Sanity check code

#Load album covers – Could have scraped them from Spotify, however, problems with Spotify’s authentication made me lose some development time

RiriTalkThatTalk <- load.image("C:\\Users\\kirka\\Documents\\Saint\\Regis\\Data Science Practicum II - MSDS 696 X 70\\Assignment\\RiriTalkThatTalk.jpg")

RiriAGirlLikeMe <- load.image("C:\\Users\\kirka\\Documents\\Saint\\Regis\\Data Science Practicum II - MSDS 696 X 70\\Assignment\\RiriAGirlLikeMe.jpg")

RiriRatedR <- load.image("C:\\Users\\kirka\\Documents\\Saint\\Regis\\Data Science Practicum II - MSDS 696 X 70\\Assignment\\RiriRatedR.jpg")

RiriMusicOfTheSun <- load.image("C:\\Users\\kirka\\Documents\\Saint\\Regis\\Data Science Practicum II - MSDS 696 X 70\\Assignment\\RiriMusicOfTheSun.jpg")

RiriGoodGirlGoneBad <- load.image("C:\\Users\\kirka\\Documents\\Saint\\Regis\\Data Science Practicum II - MSDS 696 X 70\\Assignment\\RiriGoodGirlGoneBad.jpg")

RiriUnapologetic <- load.image("C:\\Users\\kirka\\Documents\\Saint\\Regis\\Data Science Practicum II - MSDS 696 X 70\\Assignment\\RiriUnapologetic.jpg")

RiriLoud <- load.image("C:\\Users\\kirka\\Documents\\Saint\\Regis\\Data Science Practicum II - MSDS 696 X 70\\Assignment\\RiriLoud.jpg")

RiriLoud <- load.image("C:\\Users\\kirka\\Documents\\Saint\\Regis\\Data Science Practicum II - MSDS 696 X 70\\Assignment\\RiriNeededMe.jpg")

# “like” and “Like” are in the data frame

ririSongAnalysis <- within(ririSongAnalysis, Album[Album == "A Girl like Me"] <- "A Girl Like Me")

#Store album covers in dataframe

ririSongAnalysis <-

mutate(ririSongAnalysis, AlbumPic =

ifelse(Album == "Talk That Talk", RiriTalkThatTalk,

ifelse(Album == "A Girl Like Me", RiriAGirlLikeMe,

ifelse(Album == "Rated R", RiriRatedR,

ifelse(Album == "Music Of The Sun", RiriMusicOfTheSun,

ifelse(Album == "Good Girl Gone Bad", RiriGoodGirlGoneBad,

ifelse(Album == "Unapologetic", RiriUnapologetic,

ifelse(Album == "Loud", RiriLoud,

ifelse(Album == "needed me", RiriAnti,

NA)))))))))

#Multiply anger percentage by 100 to show it better on the graph

ririSongAnalysis$AngerCalc <- ririSongAnalysis$AngerPct \* 100

#Calculate anger in every album. Get the average anger per album. Album release date is plotted with the songs. Finally, a tooltip was added.

plot\_df <- ririSongAnalysis %>%

rowwise %>%

mutate(tooltip = paste0('<a style = "margin-right:', max(max(nchar(Song), nchar(Album)) \* 7, 55), 'px">', # dynamic sizing

'<img src=', AlbumPic, ' height="50" style="float:left;margin-right:5px">',

'<b>Album:</b> ', Album,

'<br><b>Song:</b> ', Song)) %>%

ungroup

avg\_line <- plot\_df %>%

group\_by(Year, Album, AlbumPic) %>%

summarise(avg = mean(AngerCalc)) %>%

ungroup %>%

transmute(x = as.numeric(as.factor(Year)),

y = avg,

tooltip = paste0('<a style = "margin-right:55px">',

'<img src=', AlbumPic, ' height="50" style="float:left;margin-right:5px">',

'<b>Album:</b> ', Album,

'<br><b>Average Anger:</b> ', round(avg, 2),

'</a>'))

plot\_ririSongAnalysis <- plot\_df %>%

mutate(tooltip = paste0(tooltip, '<br><b>Anger Index:</b> ', AngerCalc, '</a>'),

album\_number = as.numeric(as.factor(Year))) %>%

ungroup

album\_chart <- hchart(plot\_ririSongAnalysis, 'scatter', hcaes(x = as.numeric(as.factor(Year)), y = AngerCalc, group = Album)) %>%

hc\_add\_series(data = avg\_line, type = 'line') %>%

hc\_tooltip(formatter = JS(paste0("function() {return this.point.tooltip;}")), useHTML = T) %>%

hc\_colors(c(sample(brewer.pal(n\_distinct(ririSongAnalysis$Album), 'Paired')), 'black')) %>%

hc\_xAxis(title = list(text = 'Album'), labels = list(enabled = F)) %>%

hc\_yAxis(max = 30, title = list(text = 'Anger Index')) %>%

hc\_title(text = 'Anger Analysis') %>%

hc\_subtitle(text = 'Anger in songs by Rihanna') %>%

hc\_add\_theme(hc\_theme\_smpl())

album\_chart$x$hc\_opts$series[[10]]$name <- 'Avg per Album'

album\_chart