HR Analytics - Amazon and Google Reviews

Suyash Tiwari

01/13/18

Table of Contents

#Text Mining Case Study:  
#----Loading all Required Packages---------------------------------  
library("qdap")

## Warning: package 'qdap' was built under R version 3.4.3

## Loading required package: qdapDictionaries

## Loading required package: qdapRegex

## Warning: package 'qdapRegex' was built under R version 3.4.3

## Loading required package: qdapTools

## Warning: package 'qdapTools' was built under R version 3.4.3

## Loading required package: RColorBrewer

##   
## Attaching package: 'qdap'

## The following object is masked from 'package:base':  
##   
## Filter

library("tm")

## Warning: package 'tm' was built under R version 3.4.3

## Loading required package: NLP

##   
## Attaching package: 'NLP'

## The following object is masked from 'package:qdap':  
##   
## ngrams

##   
## Attaching package: 'tm'

## The following objects are masked from 'package:qdap':  
##   
## as.DocumentTermMatrix, as.TermDocumentMatrix

library("wordcloud")

## Warning: package 'wordcloud' was built under R version 3.4.3

library("dendextend")

## Warning: package 'dendextend' was built under R version 3.4.3

##   
## ---------------------  
## Welcome to dendextend version 1.6.0  
## Type citation('dendextend') for how to cite the package.  
##   
## Type browseVignettes(package = 'dendextend') for the package vignette.  
## The github page is: https://github.com/talgalili/dendextend/  
##   
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues  
## Or contact: <tal.galili@gmail.com>  
##   
## To suppress this message use: suppressPackageStartupMessages(library(dendextend))  
## ---------------------

##   
## Attaching package: 'dendextend'

## The following object is masked from 'package:qdap':  
##   
## %>%

## The following object is masked from 'package:stats':  
##   
## cutree

library("RColorBrewer")  
library("plotrix")

## Warning: package 'plotrix' was built under R version 3.4.3

library("RWeka")

## Warning: package 'RWeka' was built under R version 3.4.3

library("readr")

## Warning: package 'readr' was built under R version 3.4.2

#------1. Load readr package and Read csv files of Amazon and Google  
  
amzn <- read\_csv("C:/Users/suyas/Downloads/TwitteR/Datacamp/Text Mining/Amazon.csv")

## Parsed with column specification:  
## cols(  
## pg\_num = col\_integer(),  
## url = col\_character(),  
## pros = col\_character(),  
## cons = col\_character()  
## )

goog <- read\_csv("C:/Users/suyas/Downloads/TwitteR/Datacamp/Text Mining/google.csv")

## Parsed with column specification:  
## cols(  
## pg\_num = col\_integer(),  
## url = col\_character(),  
## pros = col\_character(),  
## cons = col\_character()  
## )

head(goog)

## # A tibble: 6 x 4  
## pg\_num url  
## <int> <chr>  
## 1 1 https://www.glassdoor.com/Reviews/Google-Reviews-E9079\_P1.htm  
## 2 1 https://www.glassdoor.com/Reviews/Google-Reviews-E9079\_P1.htm  
## 3 1 https://www.glassdoor.com/Reviews/Google-Reviews-E9079\_P1.htm  
## 4 1 https://www.glassdoor.com/Reviews/Google-Reviews-E9079\_P1.htm  
## 5 1 https://www.glassdoor.com/Reviews/Google-Reviews-E9079\_P1.htm  
## 6 1 https://www.glassdoor.com/Reviews/Google-Reviews-E9079\_P1.htm  
## # ... with 2 more variables: pros <chr>, cons <chr>

#-----2. Examine the text sources-------  
# Print the structure of amzn  
str(amzn)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 500 obs. of 4 variables:  
## $ pg\_num: int 50 50 50 50 50 50 50 50 50 50 ...  
## $ url : chr "https://www.glassdoor.com/Reviews/Amazon-com-Reviews-E6036\_P50.htm" "https://www.glassdoor.com/Reviews/Amazon-com-Reviews-E6036\_P50.htm" "https://www.glassdoor.com/Reviews/Amazon-com-Reviews-E6036\_P50.htm" "https://www.glassdoor.com/Reviews/Amazon-com-Reviews-E6036\_P50.htm" ...  
## $ pros : chr "You're surrounded by smart people and the projects are interesting, if a little daunting." "Brand name is great. Have yet to meet somebody who is unfamiliar with Amazon. Hours weren't as bad as I had pre"| \_\_truncated\_\_ "Good money.Interaction with some great minds in the world during internal conferences and sessions.Of course th"| \_\_truncated\_\_ "nice pay and overtime and different shifts" ...  
## $ cons : chr "Internal tools proliferation has created a mess for trying to get to basic information. Most people are require"| \_\_truncated\_\_ "not the most stimulating work. Good brand name to work for but the work itself is mundane as it can get. As a f"| \_\_truncated\_\_ "No proper growth plan for employees.Difficult promotion process requiring a lot more documentation than your ac"| \_\_truncated\_\_ "didn't last quite long enough" ...  
## - attr(\*, "spec")=List of 2  
## ..$ cols :List of 4  
## .. ..$ pg\_num: list()  
## .. .. ..- attr(\*, "class")= chr "collector\_integer" "collector"  
## .. ..$ url : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_character" "collector"  
## .. ..$ pros : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_character" "collector"  
## .. ..$ cons : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_character" "collector"  
## ..$ default: list()  
## .. ..- attr(\*, "class")= chr "collector\_guess" "collector"  
## ..- attr(\*, "class")= chr "col\_spec"

# Create amzn\_pros  
amzn\_pros <- amzn$pros  
  
# Create amzn\_cons  
amzn\_cons <- amzn$cons  
  
# Print the structure of goog  
str(goog)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 501 obs. of 4 variables:  
## $ pg\_num: int 1 1 1 1 1 1 1 1 1 1 ...  
## $ url : chr "https://www.glassdoor.com/Reviews/Google-Reviews-E9079\_P1.htm" "https://www.glassdoor.com/Reviews/Google-Reviews-E9079\_P1.htm" "https://www.glassdoor.com/Reviews/Google-Reviews-E9079\_P1.htm" "https://www.glassdoor.com/Reviews/Google-Reviews-E9079\_P1.htm" ...  
## $ pros : chr "\* If you're a software engineer, you're among the kings of the hill at Google. It's an engineer-driven company "| \_\_truncated\_\_ "1) Food, food, food. 15+ cafes on main campus (MTV) alone. Mini-kitchens, snacks, drinks, free breakfast/lunch/"| \_\_truncated\_\_ "You can't find a more well-regarded company that actually deserves the hype it gets." "#NAME?" ...  
## $ cons : chr "\* It \*is\* becoming larger, and with it comes growing pains: bureaucracy, slow to respond to market threats, blo"| \_\_truncated\_\_ "1) Work/life balance. What balance? All those perks and benefits are an illusion. They keep you at work and the"| \_\_truncated\_\_ "I live in SF so the commute can take between 1.5 hours to 1.75 hours each way on the shuttle - sometimes 2 hour"| \_\_truncated\_\_ "- Google is a big company. So there are going to be winners and losers when it comes to career growth. Due to t"| \_\_truncated\_\_ ...  
## - attr(\*, "spec")=List of 2  
## ..$ cols :List of 4  
## .. ..$ pg\_num: list()  
## .. .. ..- attr(\*, "class")= chr "collector\_integer" "collector"  
## .. ..$ url : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_character" "collector"  
## .. ..$ pros : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_character" "collector"  
## .. ..$ cons : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_character" "collector"  
## ..$ default: list()  
## .. ..- attr(\*, "class")= chr "collector\_guess" "collector"  
## ..- attr(\*, "class")= chr "col\_spec"

# Create goog\_pros  
goog\_pros <- goog$pros  
  
  
# Create goog\_cons  
goog\_cons <- goog$cons  
  
#---------3. Text Organization-------  
  
#qdap Cleaning function  
qdap\_clean <- function(x){  
 x <- replace\_abbreviation(x)  
 x <- replace\_contraction(x)  
 x <- replace\_number(x)  
 x <- replace\_ordinal(x)  
 x <- replace\_symbol(x)  
 x <- tolower(x)  
 return(x)  
}  
  
# tm cleaning function  
tm\_clean <- function(corpus){  
 corpus <- tm\_map(corpus, removePunctuation)  
 corpus <- tm\_map(corpus, stripWhitespace)  
 corpus <- tm\_map(corpus, removeWords, c(stopwords("en"),"Google","Amazon","company"))  
 return(corpus)  
}  
  
  
#Clean up Amazon Reviews  
# Alter amzn\_pros  
amzn\_pros <- qdap\_clean(amzn\_pros)  
amzn\_pros <- na.omit(amzn\_pros)  
  
# Alter amzn\_cons  
amzn\_cons <- qdap\_clean(amzn\_cons)  
amzn\_cons <- na.omit(amzn\_cons)  
  
# Create az\_p\_corp   
az\_p\_corp <- VCorpus(VectorSource(amzn\_pros))  
  
# Create az\_c\_corp  
az\_c\_corp <- VCorpus(VectorSource(amzn\_cons))  
  
# Create amzn\_pros\_corp  
amzn\_pros\_corp <- tm\_clean(az\_p\_corp)  
  
# Create amzn\_cons\_corp  
amzn\_cons\_corp <- tm\_clean(az\_c\_corp)  
  
# Clean up Google Reviews  
# Apply qdap\_clean to goog\_pros  
goog\_pros <- qdap\_clean(goog\_pros)  
goog\_pros <- na.omit(goog\_pros)  
  
# Apply qdap\_clean to goog\_cons  
goog\_cons <- qdap\_clean(goog\_cons)  
goog\_cons <- na.omit(goog\_cons)  
  
# Create goog\_p\_corp  
goog\_p\_corp <- VCorpus(VectorSource(goog\_pros))  
  
# Create goog\_c\_corp  
goog\_c\_corp <- VCorpus(VectorSource(goog\_cons))  
  
# Create goog\_pros\_corp  
goog\_pros\_corp <- tm\_clean(goog\_p\_corp)  
  
# Create goog\_cons\_corp  
goog\_cons\_corp <- tm\_clean(goog\_c\_corp)  
  
  
#---4 & 5. Feature Extraction and Analysis----  
  
#define tokenizer function to create bigrams:  
tokenizer <- function(x)  
 NGramTokenizer(x, Weka\_control(min = 2, max = 2))  
  
#--Word Cloud of Amazon Pros Comment---  
# Create amzn\_p\_tdm  
amzn\_p\_tdm <- TermDocumentMatrix(amzn\_pros\_corp, control = list(tokenize = tokenizer))  
  
# Create amzn\_p\_tdm\_m  
amzn\_p\_tdm\_m <- as.matrix(amzn\_p\_tdm)  
  
  
# Create amzn\_p\_freq  
amzn\_p\_freq <- rowSums(amzn\_p\_tdm\_m)  
  
# Plot a wordcloud using amzn\_p\_freq values  
wordcloud(names(amzn\_p\_freq),amzn\_p\_freq, max.words = 25, color = "blue")

## Warning in wordcloud(names(amzn\_p\_freq), amzn\_p\_freq, max.words = 25, color  
## = "blue"): smart people could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(amzn\_p\_freq), amzn\_p\_freq, max.words = 25, color  
## = "blue"): unpaid time could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(amzn\_p\_freq), amzn\_p\_freq, max.words = 25, color  
## = "blue"): pay benefits could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(amzn\_p\_freq), amzn\_p\_freq, max.words = 25,  
## color = "blue"): work environment could not be fit on page. It will not be  
## plotted.

## Warning in wordcloud(names(amzn\_p\_freq), amzn\_p\_freq, max.words = 25, color  
## = "blue"): great benefits could not be fit on page. It will not be plotted.

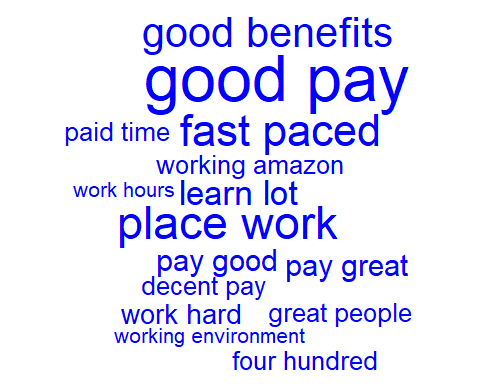
## Warning in wordcloud(names(amzn\_p\_freq), amzn\_p\_freq, max.words = 25, color  
## = "blue"): great pay could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(amzn\_p\_freq), amzn\_p\_freq, max.words = 25, color  
## = "blue"): good work could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(amzn\_p\_freq), amzn\_p\_freq, max.words = 25, color  
## = "blue"): great work could not be fit on page. It will not be plotted.

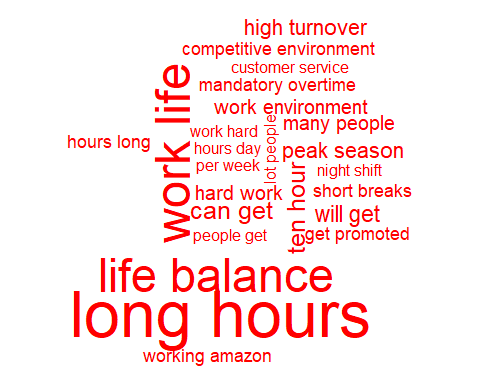
## Warning in wordcloud(names(amzn\_p\_freq), amzn\_p\_freq, max.words = 25, color  
## = "blue"): great place could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(amzn\_p\_freq), amzn\_p\_freq, max.words = 25, color  
## = "blue"): people work could not be fit on page. It will not be plotted.

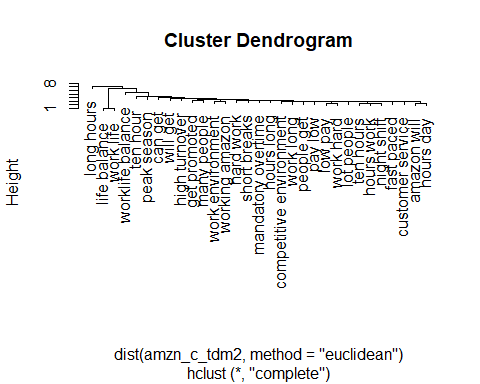


#--Word Cloud of Amazon Cons Comment---  
# Create amzn\_c\_tdm  
amzn\_c\_tdm <- TermDocumentMatrix(amzn\_cons\_corp, control = list(tokenize = tokenizer))  
  
# Create amzn\_c\_tdm\_m  
amzn\_c\_tdm\_m <- as.matrix(amzn\_c\_tdm)  
  
# Create amzn\_c\_freq   
amzn\_c\_freq <- rowSums(amzn\_c\_tdm\_m)  
  
# Plot a wordcloud of negative Amazon bigrams  
wordcloud(names(amzn\_c\_freq), amzn\_c\_freq, max.words = 25, color = "red")

## Warning in wordcloud(names(amzn\_c\_freq), amzn\_c\_freq, max.words = 25,  
## color = "red"): worklife balance could not be fit on page. It will not be  
## plotted.



#----Dendrogram for Amazon Cons-------------------  
  
# Create amzn\_c\_tdm2 by removing sparse terms   
amzn\_c\_tdm2 <- removeSparseTerms(amzn\_c\_tdm , sparse = 0.993)  
  
# Create hc as a cluster of distance values  
hc <- hclust(dist(amzn\_c\_tdm2, method = "euclidean"), method = "complete")  
  
# Produce a plot of hc  
plot(hc)



#----Word Association for Amazon Pros--------  
  
# Create term\_frequency for Amazon Pros Reviews sorted in descending order  
amz\_p\_term\_frequency <- sort(amzn\_p\_freq , decreasing = TRUE)  
   
# Print the 5 most common terms  
print(amz\_p\_term\_frequency[1:5])

## good pay great benefits smart people place work fast paced   
## 25 24 20 17 16

# Find associations with fast paced  
findAssocs(amzn\_p\_tdm,"fast paced", 0.2)

## $`fast paced`  
## paced environment environments ever learn fast   
## 0.49 0.35 0.35   
## paced friendly paced work able excel   
## 0.35 0.35 0.25   
## activity ample advance one also well   
## 0.25 0.25 0.25   
## amazon fast amazon noting amazon one   
## 0.25 0.25 0.25   
## amount time ample opportunity assistance ninety   
## 0.25 0.25 0.25   
## benefits including break computer call activity   
## 0.25 0.25 0.25   
## can choose catchy cheers center things   
## 0.25 0.25 0.25   
## challenging expect cheers opportunity choose success   
## 0.25 0.25 0.25   
## combined encouragement competitive environments computer room   
## 0.25 0.25 0.25   
## cool things deliver results dock makes   
## 0.25 0.25 0.25   
## driven deliver easy learn emphasis shipping   
## 0.25 0.25 0.25   
## encouragement innovation environment benefits environment catchy   
## 0.25 0.25 0.25   
## environment center environment fast environment help   
## 0.25 0.25 0.25   
## environment smart ever known ever witnessed   
## 0.25 0.25 0.25   
## everchanging fast everyones preferences excel advance   
## 0.25 0.25 0.25   
## excel everchanging exciting environment expect learn   
## 0.25 0.25 0.25   
## extremely fast facility top fail successful   
## 0.25 0.25 0.25   
## fantastic able fired part five percent   
## 0.25 0.25 0.25   
## freindly place friendly atmosphere friendly management   
## 0.25 0.25 0.25   
## full medical get fired go extremely   
## 0.25 0.25 0.25   
## great plenty great teamwork happening technology   
## 0.25 0.25 0.25   
## hassle benefits help get help workers   
## 0.25 0.25 0.25   
## high quality high volume including full   
## 0.25 0.25 0.25   
## innovation owning job requirements leader can   
## 0.25 0.25 0.25   
## line break lot responsibility maintaining high   
## 0.25 0.25 0.25   
## makes time management nice nice facility   
## 0.25 0.25 0.25   
## ninety five noting short offers opportunity   
## 0.25 0.25 0.25   
## one competitive one fast opportunity overtime   
## 0.25 0.25 0.25   
## opportunity yell ownership fast owning work   
## 0.25 0.25 0.25   
## paced emphasis paced exciting paced high   
## 0.25 0.25 0.25   
## paced never paced rewarding paced ship   
## 0.25 0.25 0.25   
## paced software paid upfront people focused   
## 0.25 0.25 0.25   
## percent paid plenty shifts position fast   
## 0.25 0.25 0.25   
## possible still preferences fast products quickly   
## 0.25 0.25 0.25   
## quality bar quickly possible readily available   
## 0.25 0.25 0.25   
## requirements easy responsibility ownership results great   
## 0.25 0.25 0.25   
## results team rewarding people shifts everyones   
## 0.25 0.25 0.25   
## ship dock shipping products short amount   
## 0.25 0.25 0.25   
## short fantastic smart coworkers still maintaining   
## 0.25 0.25 0.25   
## success fail successful also team driven   
## 0.25 0.25 0.25   
## technology today things happening things lot   
## 0.25 0.25 0.25   
## time fast time go top line   
## 0.25 0.25 0.25   
## upfront experience vision well volume call   
## 0.25 0.25 0.25   
## well rewarded well tuition witnessed combined   
## 0.25 0.25 0.25   
## work can work cool work environments   
## 0.25 0.25 0.25   
## work fast work job workers readily   
## 0.25 0.25 0.25   
## yell leader   
## 0.25

#----Comparison Cloud for Google Pros and Cons Reviews------  
all\_pros\_goog <- paste(goog\_pros, collapse = " ")  
all\_cons\_goog <- paste(goog\_cons, collapse = " ")  
  
all\_goog\_reviews <- c(all\_pros\_goog,all\_cons\_goog)  
  
# Create Corpus and Clean all google reviews:  
all\_goog\_reviews\_source <- VectorSource(all\_goog\_reviews)  
all\_goog\_corp <- VCorpus(all\_goog\_reviews\_source)  
all\_goog\_corp <- tm\_clean(all\_goog\_corp)  
  
# Create TDM of all Google Reviews  
all\_goog\_tdm <- TermDocumentMatrix(all\_goog\_corp)  
  
# Name the columns of all\_goog\_tdm  
colnames(all\_goog\_tdm) <- c("Goog\_Pros", "Goog\_Cons")  
  
  
# Create all\_goog\_m  
all\_goog\_m <- as.matrix(all\_goog\_tdm)  
  
# Build a comparison cloud  
comparison.cloud(all\_goog\_m, colors = c("#F44336", "#2196f3"), max.words = 100)

## Warning in comparison.cloud(all\_goog\_m, colors = c("#F44336", "#2196f3"), :  
## awesome could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(all\_goog\_m, colors = c("#F44336", "#2196f3"), :  
## interesting could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(all\_goog\_m, colors = c("#F44336", "#2196f3"), :  
## opportunities could not be fit on page. It will not be plotted.

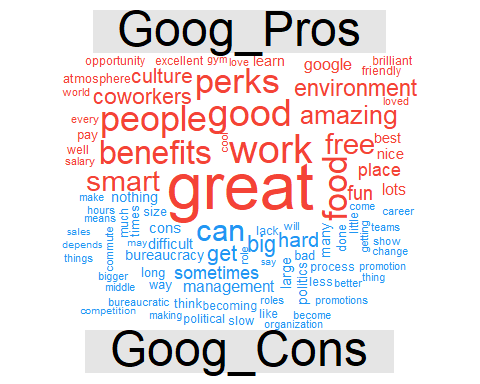
## Warning in comparison.cloud(all\_goog\_m, colors = c("#F44336", "#2196f3"), :  
## compensation could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(all\_goog\_m, colors = c("#F44336", "#2196f3"), :  
## exciting could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(all\_goog\_m, colors = c("#F44336", "#2196f3"), :  
## flexible could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(all\_goog\_m, colors = c("#F44336", "#2196f3"), :  
## number could not be fit on page. It will not be plotted.

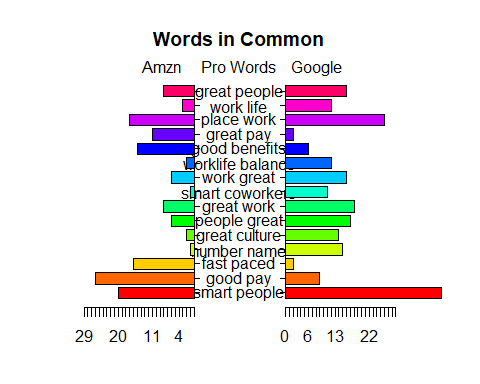
## Warning in comparison.cloud(all\_goog\_m, colors = c("#F44336", "#2196f3"), :  
## balance could not be fit on page. It will not be plotted.



#------Pyramid Cloud for Pros of Amazon and Google Reviews-----  
  
all\_pros\_amzn <- paste(amzn\_pros, collapse = " ")  
all\_pros\_goog <- paste(goog\_pros, collapse = " ")  
  
all\_pros\_reviews <- c(all\_pros\_amzn,all\_pros\_goog)  
  
# Create Corpus and Clean all google and amazon pros reviews:  
all\_pros\_reviews\_source <- VectorSource(all\_pros\_reviews)  
all\_pros\_corp <- VCorpus(all\_pros\_reviews\_source)  
all\_pros\_corp <- tm\_clean(all\_pros\_corp)  
  
# Create TDM of all Google and Amazon pros Reviews  
all\_pros\_tdm <- TermDocumentMatrix(all\_pros\_corp, control = list(tokenize = tokenizer))  
  
# Name the columns of all\_pros\_tdm  
colnames(all\_pros\_tdm) <- c("Amazon Pro", "Google Pro")  
  
  
# Create all\_pros\_tdm\_m  
all\_pros\_tdm\_m <- as.matrix(all\_pros\_tdm)  
  
head(all\_pros\_tdm\_m)

## Docs  
## Terms Amazon Pro Google Pro  
## <U+0085> show 0 1  
## ability customer 1 0  
## ability iterate 1 0  
## ability make 1 1  
## ability see 1 0  
## ability switch 0 1

# Create common\_words  
common\_words <- subset(all\_pros\_tdm\_m, all\_pros\_tdm\_m[, 1] > 0 & all\_pros\_tdm\_m[, 2] > 0)  
  
# Create difference  
difference <- abs(common\_words[,1] - common\_words[,2] )  
  
# Add difference to common\_words  
common\_words <- cbind(common\_words,difference)  
  
# Order the data frame from most differences to least  
common\_words <- common\_words[order(common\_words[, 3], decreasing = TRUE),]  
  
# Create top15\_df  
top15\_df <- data.frame(x = common\_words[1:15, 1],   
 y = common\_words[1:15, 2],   
 labels = rownames(common\_words[1:15, ]))  
  
# Create the pyramid plot  
pyramid.plot(top15\_df$x, top15\_df$y, labels = top15\_df$labels, gap = 12,   
 top.labels = c("Amzn", "Pro Words", "Google"),   
 main = "Words in Common", unit = NULL)

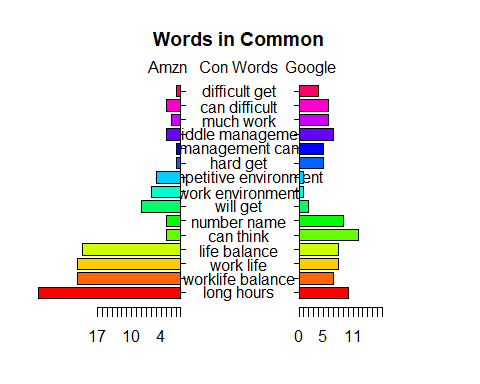


## [1] 5.1 4.1 4.1 2.1

#------Pyramid Cloud for Cons of Amazon and Google Reviews-----  
  
all\_cons\_amzn <- paste(amzn\_cons, collapse = " ")  
all\_cons\_goog <- paste(goog\_cons, collapse = " ")  
  
all\_cons\_reviews <- c(all\_cons\_amzn,all\_cons\_goog)  
  
# Create Corpus and Clean all google and amazon pros reviews:  
all\_cons\_reviews\_source <- VectorSource(all\_cons\_reviews)  
all\_cons\_corp <- VCorpus(all\_cons\_reviews\_source)  
all\_cons\_corp <- tm\_clean(all\_cons\_corp)  
  
# Create TDM of all Google and Amazon pros Reviews  
all\_cons\_tdm <- TermDocumentMatrix(all\_cons\_corp, control = list(tokenize = tokenizer))  
  
# Name the columns of all\_pros\_tdm  
colnames(all\_cons\_tdm) <- c("Amazon Con", "Google Con")  
  
  
# Create all\_pros\_tdm\_m  
all\_cons\_tdm\_m <- as.matrix(all\_cons\_tdm)  
  
head(all\_cons\_tdm\_m)

## Docs  
## Terms Amazon Con Google Con  
## <U+0085> show 0 5  
## abandon open 0 1  
## ability advance 1 0  
## able sacrifice 1 0  
## able sit 1 0  
## able work 1 1

# Create common\_words  
common\_words\_cons <- subset(all\_cons\_tdm\_m, all\_cons\_tdm\_m[, 1] > 0 & all\_cons\_tdm\_m[, 2] > 0)  
  
# Create difference  
difference\_cons <- abs(common\_words\_cons[,1] - common\_words\_cons[,2] )  
  
# Add difference to common\_words  
common\_words\_cons <- cbind(common\_words\_cons,difference\_cons)  
  
# Order the data frame from most differences to least  
common\_words\_cons <- common\_words\_cons[order(common\_words\_cons[, 3], decreasing = TRUE),]  
  
# Create top15\_df  
top15\_df\_cons <- data.frame(x = common\_words\_cons[1:15, 1],   
 y = common\_words\_cons[1:15, 2],   
 labels = rownames(common\_words\_cons[1:15, ]))  
  
# Create the pyramid plot  
pyramid.plot(top15\_df\_cons$x, top15\_df\_cons$y, labels = top15\_df\_cons$labels, gap = 12,   
 top.labels = c("Amzn", "Con Words", "Google"),   
 main = "Words in Common", unit = NULL)



## [1] 4 2 4 2