Final Project RIT

Saeed

May 10, 2019

Table of Contents



## Load required libraries

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

## Loading required package: RColorBrewer

##   
## Attaching package: 'syuzhet'

## The following object is masked from 'package:rtweet':  
##   
## get\_tokens

##   
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':  
##   
## last\_plot

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

## The following object is masked from 'package:graphics':  
##   
## layout

##   
## Attaching package: 'igraph'

## The following object is masked from 'package:plotly':  
##   
## groups

## The following objects are masked from 'package:dplyr':  
##   
## as\_data\_frame, groups, union

## The following objects are masked from 'package:stats':  
##   
## decompose, spectrum

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

## Authenticate to the Twitter Rest API

consumerKey <- "\*\*\*"  
consumerSecret <- "\*\*\*"  
accessToken <- "\*\*\*"  
accessTokenSecret <- "\*\*\*"  
#Authenticate to the Twitter Rest API   
create\_token(app = "Kafka Course Starter", consumerKey, consumerSecret,  
 access\_token = accessToken, access\_secret = accessTokenSecret, set\_renv = TRUE)

## Search for the relevant tag for your analysis

#Search for the relevant tag for your analysis  
azure\_tweets\_search<-search\_tweets("#Azure OR #azurecloud OR azure", n = 1020, lang='en', type = "mixed", include\_rts = FALSE,  
 geocode =NULL, max\_id = NULL, parse = TRUE, token = NULL,verbose = TRUE)  
aws\_tweets\_search<-search\_tweets("#AWS OR #awscloud OR amazoncloud", n = 1300, lang='en', type = "mixed", include\_rts = FALSE,  
 geocode =NULL, max\_id = NULL, parse = TRUE, token = NULL, verbose = TRUE)  
google\_tweets\_search<-search\_tweets("#GoogleCloud OR #GCPCloud OR googlecloud", n = 1020, lang='en', type = "mixed", include\_rts = FALSE,  
 geocode =NULL, max\_id = NULL, parse = TRUE, token = NULL, verbose = TRUE)

## Selecting the needed columns for the analysis

#Selecting the needed columns for our research  
azure\_tweets\_search<-data.frame(azure\_tweets\_search[,c('created\_at','screen\_name','location','text')])  
aws\_tweets\_search<-data.frame(aws\_tweets\_search[,c('created\_at','screen\_name','location','text')])  
google\_tweets\_search<-data.frame(google\_tweets\_search[,c('created\_at','screen\_name','location','text')])

## Save the tweets to csv file

#save the tweets to csv file  
write.csv(azure\_tweets\_search, file = "azure\_tweets.csv")  
write.csv(aws\_tweets\_search, file = "aws\_tweets.csv")  
write.csv(google\_tweets\_search, file = "google\_tweets.csv")

## Read the tweets from csv file

#Read the tweets from csv file  
#Note: Datasets must be in your working directory   
#Set working directory eg: setwd("Your Directory")  
azure\_tweets <- read.csv("azure\_tweets.csv")  
aws\_tweets <- read.csv("aws\_tweets.csv")  
google\_tweets <- read.csv("google\_tweets.csv")  
  
  
# Returns the first item of the tweets before cleaning  
head(azure\_tweets$text,1)

## [1] Good read. Thank you @satyanadella for your vision in expanding @Microsoft's partnership with @DellTech to deliver a fully native, supported, and certified @VMware cloud infrastructure @Azure.\nhttps://t.co/ESIJLr2Yun  
## 1097 Levels: "...I'm excited to see how companies are using #tech to better serve their customers. The @StarbucksUK 'bean-to-cup' initiative is a powerful example of this..." @Microsoft's @JP\_Courtois on #Starbucks' use of @Azure, #IoT and #blockchain. Read more: https://t.co/qmlfoWfjGc https://t.co/la6moRgf59 ...

head(aws\_tweets$text,1)

## [1] Amazon S3 Path Deprecation Plan - The Rest of the Story - https://t.co/0C3epGdui7 #AWS https://t.co/AchjydsQWz  
## 843 Levels: 'Rackspace have worked with our DevOps engineers to look at our current infrastructure and ways in which we can optimise that infrastructure to be more efficient.'\n\nhttps://t.co/NeSpzz7uhV\n\n#AWS #Scalable #Technology #Enterprise #Intranet ...

head(google\_tweets$text,1)

## [1] Introducing Chrome Browser Cloud Management, a central location for admins to view and manage #Chrome Browser policies, settings and installed extensions. Learn more <U+2193> https://t.co/KGHB21Fsmh  
## 1094 Levels: 'Cyan'-ara to boring data sets in Sheets! Take a look at how to give your data a fresh new look, along with 26 other useful features in this article from @FastCompany: https://t.co/CvtT8xKadh @gsuite https://t.co/Pi7qxDHJ9v ...

## Clean the tweets text column and store the cleaned version in new column cleaned\_text, keep the original!

#Data Cleaning - Azure  
azure\_tweets$cleaned\_text<- gsub("http.\*","", azure\_tweets$text) # remove urls  
azure\_tweets$cleaned\_text<- gsub("\\W+"," ", azure\_tweets$cleaned\_text) # remove none words  
azure\_tweets$cleaned\_text<- gsub('#\\S+', '', azure\_tweets$cleaned\_text) ## remove any hashtag   
azure\_tweets$cleaned\_text<- gsub('@\\S+', '', azure\_tweets$cleaned\_text) ## remove people mentioned   
azure\_tweets$cleaned\_text<- gsub("\\d+", " ", azure\_tweets$cleaned\_text) # remove any digit or digits  
azure\_tweets$cleaned\_text<- gsub(' +',' ',azure\_tweets$cleaned\_text) ## remove whitespaces  
  
#Data Cleaning - AWS  
aws\_tweets$cleaned\_text<- gsub("http.\*","", aws\_tweets$text) # remove urls  
aws\_tweets$cleaned\_text<- gsub("\\W+"," ", aws\_tweets$cleaned\_text) # remove none words  
aws\_tweets$cleaned\_text<- gsub('#\\S+', '', aws\_tweets$cleaned\_text) ## remove any hashtag   
aws\_tweets$cleaned\_text<- gsub('@\\S+', '', aws\_tweets$cleaned\_text) ## remove people mentioned   
aws\_tweets$cleaned\_text<- gsub("\\d+", " ", aws\_tweets$cleaned\_text) # remove any digit or digits  
aws\_tweets$cleaned\_text<- gsub(' +',' ',aws\_tweets$cleaned\_text) ## remove whitespaces  
  
#Data Cleaning - Google  
google\_tweets$cleaned\_text<- gsub("http.\*","", google\_tweets$text) # remove urls  
google\_tweets$cleaned\_text<- gsub("\\W+"," ", google\_tweets$cleaned\_text) # remove none words  
google\_tweets$cleaned\_text<- gsub('#\\S+', '', google\_tweets$cleaned\_text) ## remove any hashtag   
google\_tweets$cleaned\_text<- gsub('@\\S+', '', google\_tweets$cleaned\_text) ## remove people mentioned   
google\_tweets$cleaned\_text<- gsub("\\d+", " ", google\_tweets$cleaned\_text) # remove any digit or digits  
google\_tweets$cleaned\_text<- gsub(' +',' ',google\_tweets$cleaned\_text) ## remove whitespaces  
  
  
# Returns the first item of the cleaned tweets   
head(azure\_tweets$cleaned\_text,1)

## [1] "Good read Thank you satyanadella for your vision in expanding Microsoft s partnership with DellTech to deliver a fully native supported and certified VMware cloud infrastructure Azure "

head(aws\_tweets$cleaned\_text,1)

## [1] "Amazon S Path Deprecation Plan The Rest of the Story "

head(google\_tweets$cleaned\_text,1)

## [1] "Introducing Chrome Browser Cloud Management a central location for admins to view and manage Chrome Browser policies settings and installed extensions Learn more U "

# Remove punctuation, convert to lowercase, add id for each tweet(split a column into tokens) - Tokenization

## Split the cleaned\_text into tokens using the tokenizers package

#Split a column into tokens using the tokenizers package  
azure\_tweets\_clean <- azure\_tweets %>%  
 dplyr::select(cleaned\_text) %>%  
 unnest\_tokens(word, cleaned\_text)  
  
#View the tokenization words  
head(azure\_tweets\_clean,2)

## word  
## 1 good  
## 1.1 read

#Split a column into tokens using the tokenizers package  
aws\_tweets\_clean <- aws\_tweets %>%  
 dplyr::select(cleaned\_text) %>%  
 unnest\_tokens(word, cleaned\_text)  
  
#View the tokenization words  
head(aws\_tweets\_clean,2)

## word  
## 1 amazon  
## 1.1 s

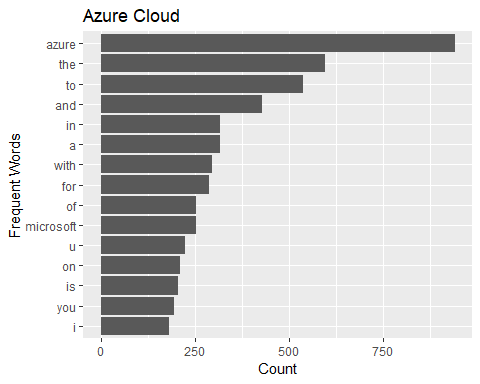
#Split a column into tokens using the tokenizers package  
google\_tweets\_clean <- google\_tweets %>%  
 dplyr::select(cleaned\_text) %>%  
 unnest\_tokens(word, cleaned\_text)  
  
#View the tokenization words   
head(google\_tweets\_clean,2)

## word  
## 1 introducing  
## 1.1 chrome

##Plot the results after cleaning

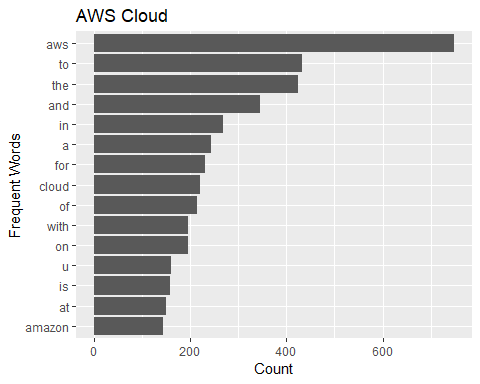
#Look for any issues by plotting the top 15 common words - Azure  
azure\_tweets\_clean %>%  
 count(word, sort = TRUE) %>%  
 top\_n(15) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(x = word, y = n))+  
 geom\_col()+ coord\_flip()+  
 labs(y = "Count",  
 x = "Frequent Words",  
 title = "Azure Cloud")

## Selecting by n



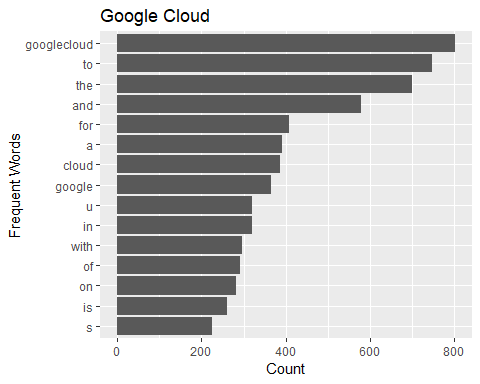
#Look for any issues by plotting the most common words - AWS  
aws\_tweets\_clean %>%  
 count(word, sort = TRUE) %>%  
 top\_n(15) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(x = word, y = n))+  
 geom\_col()+ coord\_flip()+  
 labs(y = "Count",  
 x = "Frequent Words",  
 title = "AWS Cloud")

## Selecting by n



#Look for any issues by plotting the most common words - Google  
google\_tweets\_clean %>%  
 count(word, sort = TRUE) %>%  
 top\_n(15) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(x = word, y = n))+  
 geom\_col()+ coord\_flip() +   
 labs(y = "Count",  
 x = "Frequent Words",  
 title = "Google Cloud ")

## Selecting by n



## Remove Stopwords

#View stop words from three lexicons, as a data frame  
head(stop\_words,5)

## # A tibble: 5 x 2  
## word lexicon  
## <chr> <chr>   
## 1 a SMART   
## 2 a's SMART   
## 3 able SMART   
## 4 about SMART   
## 5 above SMART

#Number of rows BEFORE removing the stop words  
nrow(azure\_tweets\_clean)

## [1] 21390

nrow(aws\_tweets\_clean)

## [1] 16814

nrow(google\_tweets\_clean)

## [1] 25074

#Remove stop words from your list of words from the three datasets  
azure\_cleaned\_tweet\_words <- azure\_tweets\_clean %>%  
 anti\_join(stop\_words)

## Joining, by = "word"

aws\_cleaned\_tweet\_words <- aws\_tweets\_clean %>%  
 anti\_join(stop\_words)

## Joining, by = "word"

google\_cleaned\_tweet\_words <- google\_tweets\_clean %>%  
 anti\_join(stop\_words)

## Joining, by = "word"

# Number of rows AFTER removing the stop words  
nrow(azure\_cleaned\_tweet\_words)

## [1] 11553

nrow(aws\_cleaned\_tweet\_words)

## [1] 9754

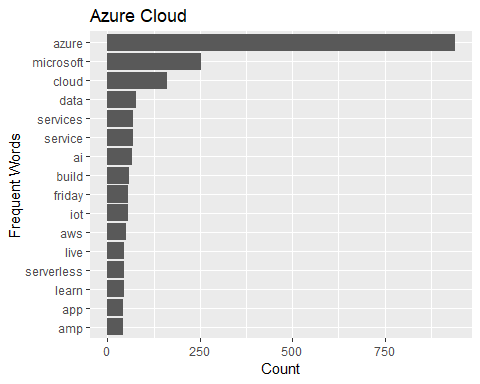
nrow(google\_cleaned\_tweet\_words)

## [1] 13267

## Plotting for further analysis and check for any remaining issues

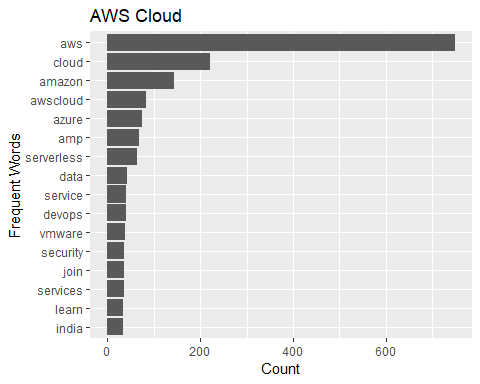
# Plot the top 15 words and check if there is any more issues in the data!   
azure\_cleaned\_tweet\_words %>%  
 count(word, sort = TRUE) %>%  
 top\_n(15) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(x = word, y = n)) +  
 geom\_col()+ coord\_flip() +   
 labs(y = "Count",  
 x = "Frequent Words",  
 title = "Azure Cloud ")

## Selecting by n



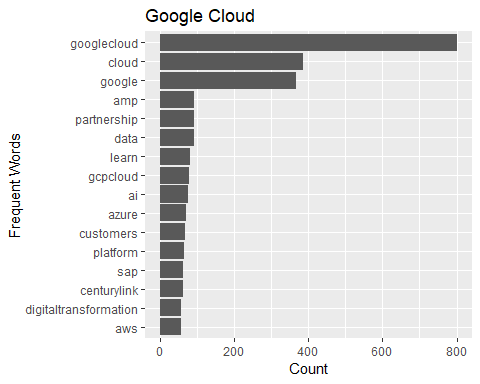
aws\_cleaned\_tweet\_words %>%  
 count(word, sort = TRUE) %>%  
 top\_n(15) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(x = word, y = n)) +  
 geom\_col()+ coord\_flip() +   
 labs(y = "Count",  
 x = "Frequent Words",  
 title = "AWS Cloud ")

## Selecting by n



google\_cleaned\_tweet\_words %>%  
 count(word, sort = TRUE) %>%  
 top\_n(15) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(x = word, y = n)) +  
 geom\_col()+ coord\_flip() +   
 labs(y = "Count",  
 x = "Frequent Words",  
 title = "Google Cloud ")

## Selecting by n



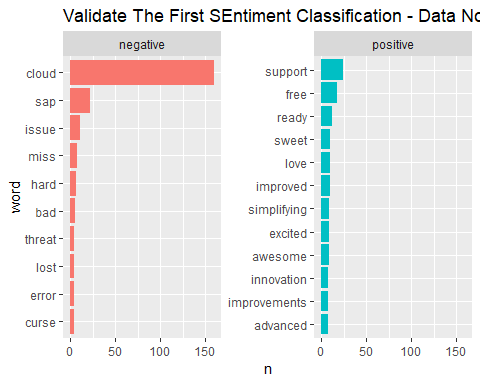
## Join with the sentiments from lexicon; either “afinn”, “bing”, “nrc” and plot the results

#Join with the sentiments from lexicon; either "afinn", "bing", "nrc"  
#Plot top 15 words to see whether its POSITIVE OR NEGATIVE   
azure\_bing\_word\_counts <- azure\_cleaned\_tweet\_words %>%  
 inner\_join(get\_sentiments("bing")) %>%  
 count(word, sentiment, sort = TRUE)

## Joining, by = "word"

azure\_bing\_word\_counts %>%  
 group\_by(sentiment) %>%  
 top\_n(10) %>%  
 ungroup() %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(word, n, fill = sentiment)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~sentiment, scales = "free\_y") +  
 labs(title = "Validate The First SEntiment Classification - Data Noise!") + coord\_flip()

## Selecting by n

 ### Issues noticed, company name repeated the most but in fact it has no meaning in terms of positive or negative Another issue is that, the word CLOUD is considered as negative which is unappropriate for our case

## Create Custom Stopwords to solve the above issues,

#Decided to add a custom stop words to handle the above issues and reduce the noise caused by them  
rit\_stop\_words <- tibble(  
 word = c(  
 "cloud","msbuild","microsoft","azure", "sap","google","googlecloud","gcp","gcpcloud","amazon","aws","awscloud"  
 ),lexicon = "tweetsrit"  
)

## Remove the words according to the custom created Stopwords

#Remove the words according to our custom created Stopwords  
azure\_cleaned\_tweet\_words <- azure\_cleaned\_tweet\_words %>%  
 anti\_join(rit\_stop\_words)

## Joining, by = "word"

aws\_cleaned\_tweet\_words <- aws\_cleaned\_tweet\_words %>%  
 anti\_join(rit\_stop\_words)

## Joining, by = "word"

google\_cleaned\_tweet\_words <- google\_cleaned\_tweet\_words %>%  
 anti\_join(rit\_stop\_words)

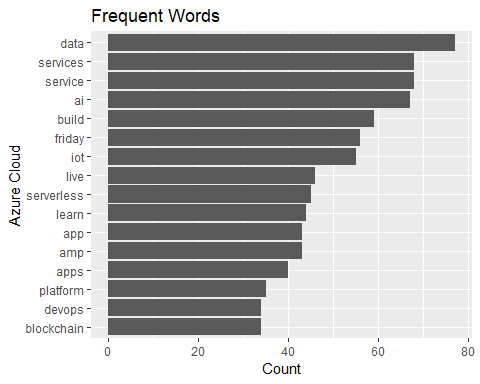
## Joining, by = "word"

## Plot the top 15 words again to see if the above issue resolved

After removing the noise - based on our custom Stopwords

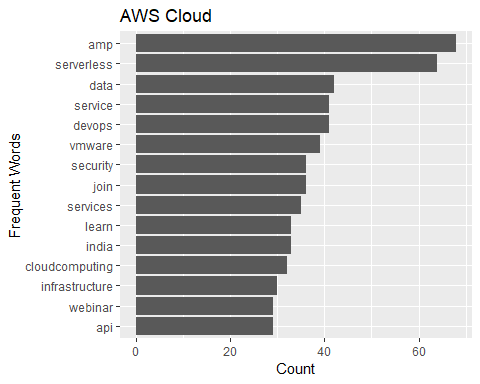
# Azure  
azure\_cleaned\_tweet\_words %>%  
 count(word, sort = TRUE) %>%  
 top\_n(15) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(x = word, y = n)) +   
 geom\_col()+ coord\_flip()+  
 labs(y = "Count",  
 x = "Azure Cloud",  
 title = "Frequent Words")

## Selecting by n



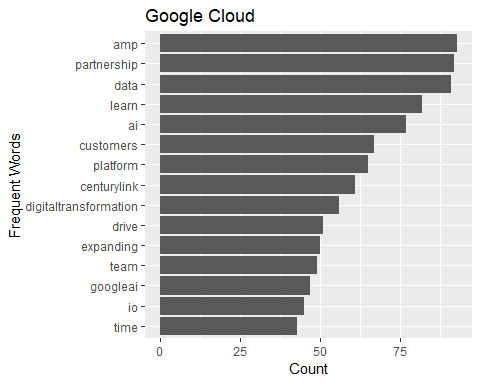
#AWS  
aws\_cleaned\_tweet\_words %>%  
 count(word, sort = TRUE) %>%  
 top\_n(15) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(x = word, y = n)) +  
 geom\_col()+ coord\_flip()+  
 labs(y = "Count",  
 title = "AWS Cloud",  
 x = "Frequent Words")

## Selecting by n



#Google  
google\_cleaned\_tweet\_words %>%  
 count(word, sort = TRUE) %>%  
 top\_n(15) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(x = word, y = n)) +   
 geom\_col()+ coord\_flip()+  
 labs(y = "Count",  
 title = "Google Cloud",  
 x = "Frequent Words")

## Selecting by n



## SENTIMENT ANALYSIS

### Join sentiment classification to the tweet words based on the three general-purpose lexicons(bing, afinn, nrc)

# Join sentiment classification to the tweet words based on the   
# three general-purpose lexicons(bing, afinn, nrc)  
  
#bing lexicon - Azure  
azure\_bing\_word\_counts <- azure\_cleaned\_tweet\_words %>%  
 inner\_join(get\_sentiments("bing")) %>%  
 count(word, sentiment, sort = TRUE)

## Joining, by = "word"

#bing lexicon - AWS  
aws\_bing\_word\_counts <- aws\_cleaned\_tweet\_words %>%  
 inner\_join(get\_sentiments("bing")) %>%  
 count(word, sentiment, sort = TRUE) %>%  
 ungroup()

## Joining, by = "word"

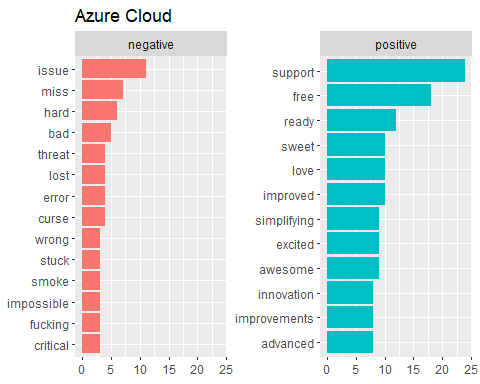
#bing lexicon - Google  
google\_bing\_word\_counts <- google\_cleaned\_tweet\_words %>%  
 inner\_join(get\_sentiments("bing")) %>%  
 count(word, sentiment, sort = TRUE) %>%  
 ungroup()

## Joining, by = "word"

## Visualize the sentiment for the cloud providers

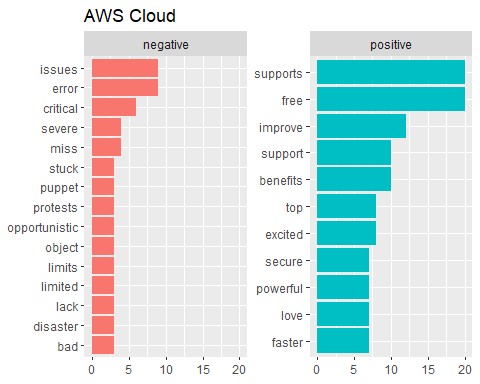
#bing lexicon  
azure\_bing\_word\_counts %>%  
 group\_by(sentiment) %>%  
 top\_n(10) %>%  
 ungroup() %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(word, n, fill = sentiment)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~sentiment, scales = "free\_y") +  
 labs(title = "Azure Cloud", x=NULL,y=NULL) + coord\_flip()

## Selecting by n



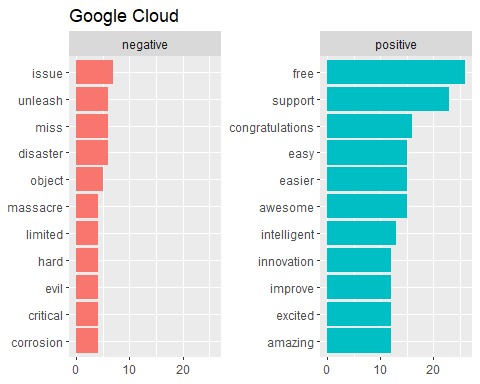
aws\_bing\_word\_counts %>%  
 group\_by(sentiment) %>%  
 top\_n(10) %>%  
 ungroup() %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(word, n, fill = sentiment)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~sentiment, scales = "free\_y") +  
 labs(title = "AWS Cloud", x=NULL,y=NULL) + coord\_flip()

## Selecting by n



google\_bing\_word\_counts %>%  
 group\_by(sentiment) %>%  
 top\_n(10) %>%  
 ungroup() %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(word, n, fill = sentiment)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~sentiment, scales = "free\_y") +  
 labs(title = "Google Cloud", x=NULL,y=NULL) + coord\_flip()

## Selecting by n



## Plotting words and classifying them into:

### Trust, Fear, Negative, Sadness, Anger, Surprise,Positive, Disgust,Joy, Anticipation

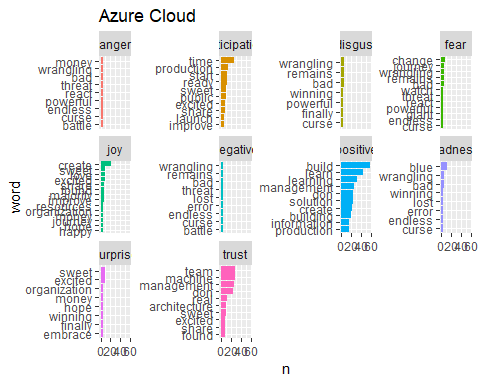
#nrc lexicon - Azure   
azure\_nrc\_word\_counts <- azure\_cleaned\_tweet\_words %>%  
 inner\_join(get\_sentiments("nrc")) %>%  
 count(word, sentiment, sort = TRUE) %>%  
 top\_n(200)%>%  
 ungroup()

## Joining, by = "word"

## Selecting by n

azure\_nrc\_word\_counts %>%  
 group\_by(sentiment) %>%  
 top\_n(10) %>%  
 ungroup() %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(word, n, fill = sentiment)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~sentiment, scales = "free\_y") +  
 labs(title = "Azure Cloud") + coord\_flip()

## Selecting by n



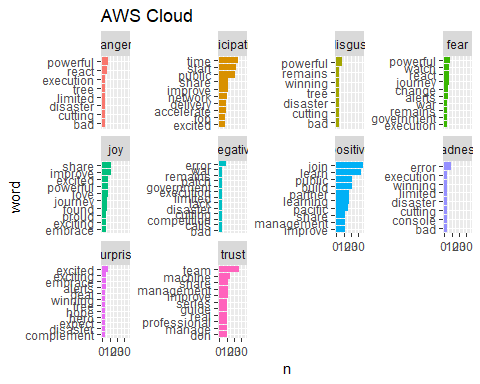
#nrc lexicon - AWS  
aws\_nrc\_word\_counts <- aws\_cleaned\_tweet\_words %>%  
 inner\_join(get\_sentiments("nrc")) %>%  
 count(word, sentiment, sort = TRUE) %>%  
 top\_n(200)%>%  
 ungroup()

## Joining, by = "word"

## Selecting by n

aws\_nrc\_word\_counts %>%  
 group\_by(sentiment) %>%  
 top\_n(10) %>%  
 ungroup() %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(word, n, fill = sentiment)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~sentiment, scales = "free\_y") +  
 labs(title = "AWS Cloud") + coord\_flip()

## Selecting by n



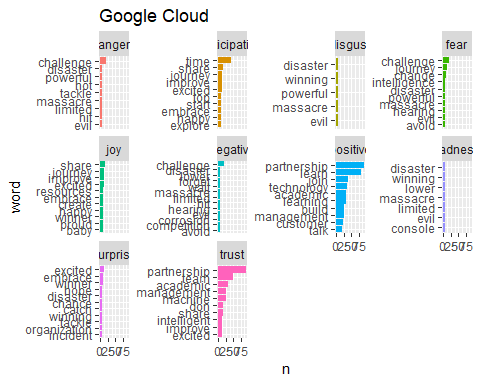
#nrc lexicon - Google  
google\_nrc\_word\_counts <- google\_cleaned\_tweet\_words %>%  
 inner\_join(get\_sentiments("nrc")) %>%  
 count(word, sentiment, sort = TRUE) %>%  
 top\_n(200)%>%  
 ungroup()

## Joining, by = "word"

## Selecting by n

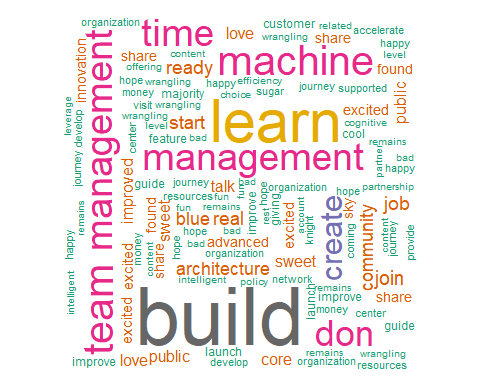
google\_nrc\_word\_counts %>%  
 group\_by(sentiment) %>%  
 top\_n(10) %>%  
 ungroup() %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(word, n, fill = sentiment)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~sentiment, scales = "free\_y") +  
 labs(title = "Google Cloud") + coord\_flip()

## Selecting by n



## Cloudwords

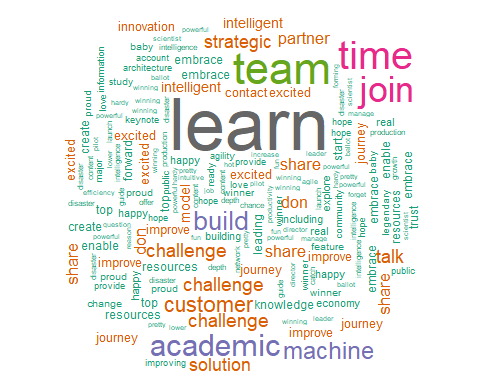
#Cloudwords PACKAGE  
#Todo: invistigate a bug in cloudword library. when previos plots in memory, the plots breaks therefore; clear the plot window  
  
#Azure  
wordcloud(azure\_nrc\_word\_counts$word,azure\_nrc\_word\_counts$n,  
 max.words = 200, min.freq = 5,colors = brewer.pal(8,"Dark2"),  
 scale = c(5,0.2), rot.per = 0.3)



#AWS  
#dev.off() #clear the plot window  
wordcloud(aws\_nrc\_word\_counts$word,aws\_nrc\_word\_counts$n,  
 max.words = 200, min.freq = 5,colors = brewer.pal(8,"Dark2"),  
 scale = c(5,0.2), rot.per = 0.3 )



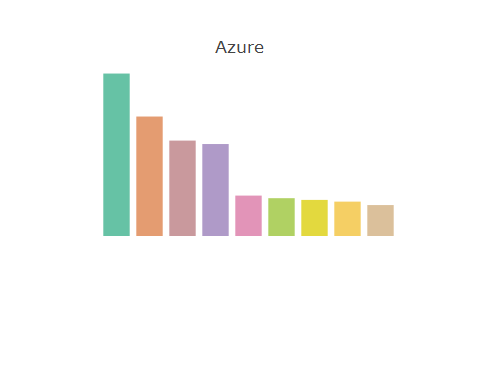
#Google  
#dev.off() #clear the plot window  
wordcloud(google\_nrc\_word\_counts$word,google\_nrc\_word\_counts$n,  
 max.words = 200, min.freq = 5,colors = brewer.pal(8,"Dark2"),  
 scale = c(5,0.2), rot.per = 0.3 )



## Get the sentiment score for each emotion - nrc

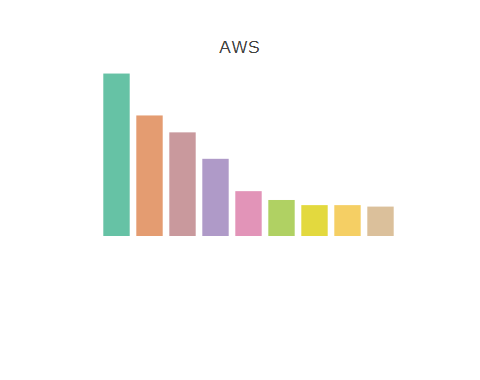
# Get the sentiment score for each emotion - nrc  
#Plot the converted values into 10 emotion levels:  
#(positive trust anticipation joy fear surprise anger disgust negative sadness)  
  
#Azure  
azure\_sentiment<-get\_nrc\_sentiment(as.vector(azure\_nrc\_word\_counts$word))  
azure\_sentiment\_df<- data.frame(count=colSums(azure\_sentiment), emotion=names(colSums(azure\_sentiment)))  
azure\_sentiment\_df$emotion = factor(azure\_sentiment\_df$emotion, levels=azure\_sentiment\_df$emotion[order(azure\_sentiment\_df$count, decreasing = TRUE)])  
plot\_ly(azure\_sentiment\_df, x=~emotion, y=~count, type="bar", color=~emotion) %>%  
 layout(xaxis=list(title=""), showlegend=FALSE, title="Azure")

## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8  
## Returning the palette you asked for with that many colors  
  
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8  
## Returning the palette you asked for with that many colors



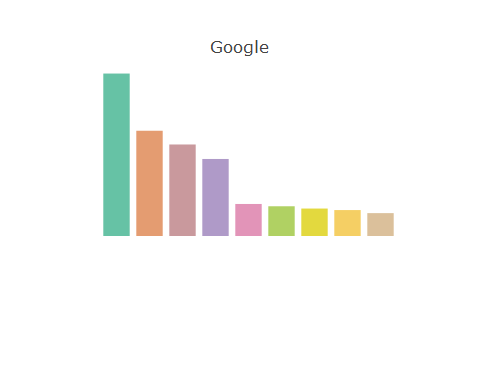
#AWS  
aws\_sentiment<-get\_nrc\_sentiment(as.vector(aws\_nrc\_word\_counts$word))  
aws\_sentiment\_df<- data.frame(count=colSums(aws\_sentiment), emotion=names(colSums(aws\_sentiment)))  
aws\_sentiment\_df$emotion = factor(aws\_sentiment\_df$emotion, levels=aws\_sentiment\_df$emotion[order(aws\_sentiment\_df$count, decreasing = TRUE)])  
plot\_ly(aws\_sentiment\_df, x=~emotion, y=~count, type="bar", color=~emotion) %>%  
 layout(xaxis=list(title=""), showlegend=FALSE, title="AWS")

## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8  
## Returning the palette you asked for with that many colors  
  
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8  
## Returning the palette you asked for with that many colors



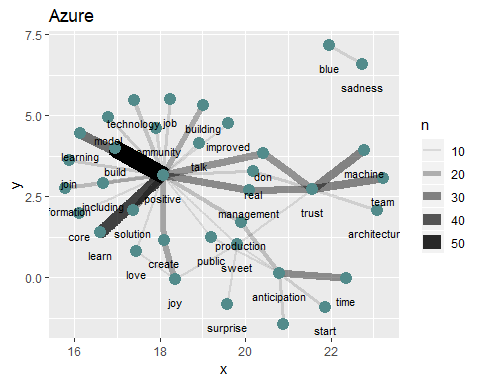
#Google  
google\_sentiment<-get\_nrc\_sentiment(as.vector(google\_nrc\_word\_counts$word))  
google\_sentiment\_df<- data.frame(count=colSums(google\_sentiment), emotion=names(colSums(google\_sentiment)))  
google\_sentiment\_df$emotion = factor(google\_sentiment\_df$emotion, levels=google\_sentiment\_df$emotion[order(google\_sentiment\_df$count, decreasing = TRUE)])  
plot\_ly(google\_sentiment\_df, x=~emotion, y=~count, type="bar", color=~emotion) %>%  
 layout(xaxis=list(title=""), showlegend=FALSE, title="Google")

## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8  
## Returning the palette you asked for with that many colors  
  
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8  
## Returning the palette you asked for with that many colors

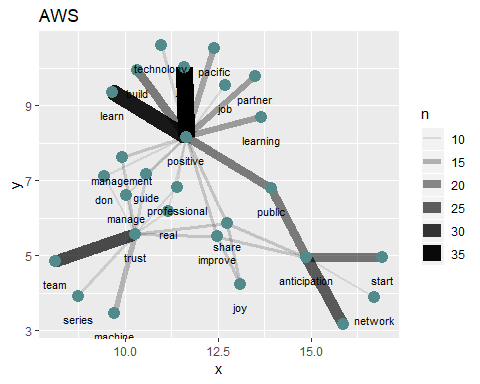


## Plot word network with minimum frequency of 10

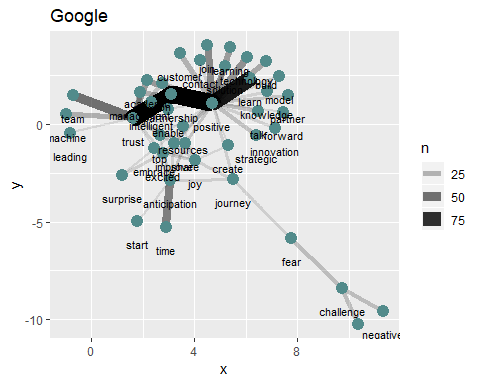
#plot word network with minimum frequency of 10  
#Azure  
azure\_nrc\_word\_counts %>%  
 filter(n >= 10) %>%  
 graph\_from\_data\_frame() %>%  
 ggraph(layout = "fr") +  
 geom\_edge\_link(aes(edge\_alpha = n, edge\_width = n)) +  
 geom\_node\_point(color = "darkslategray4", size = 4) +  
 geom\_node\_text(aes(label = name), vjust = 3, size = 3) +  
 labs(title = "Azure")



#AWS  
aws\_nrc\_word\_counts %>%  
 filter(n >= 10) %>%  
 graph\_from\_data\_frame() %>%  
 ggraph(layout = "fr") +  
 geom\_edge\_link(aes(edge\_alpha = n, edge\_width = n)) +  
 geom\_node\_point(color = "darkslategray4", size = 4) +  
 geom\_node\_text(aes(label = name), vjust = 3, size = 3) +  
 labs(title = "AWS")



#Google  
google\_nrc\_word\_counts %>%  
 filter(n >= 10) %>%  
 graph\_from\_data\_frame() %>%  
 ggraph(layout = "fr") +  
 geom\_edge\_link(aes(edge\_alpha = n, edge\_width = n)) +  
 geom\_node\_point(color = "darkslategray4", size = 4) +  
 geom\_node\_text(aes(label = name), vjust = 3, size = 3) +  
 labs(title = "Google")

 ## Thank You