Final Project RIT

Saeed

May 10, 2019

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
                            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
                                 geocode =NULL, max_id = NULL, parse = TRUE, token = NULL, verbose =
google_tweets_search<-search_tweets("#GoogleCloud OR #GCPCloud OR googlecloud", n = 1020, lang='en', ty
                                 geocode =NULL, max_id = NULL, parse = TRUE, token = NULL, verbose =
```

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] Understand the physical structure of #Azure infrastructure, redundancies, and service level agre
## 2165 Levels: '@Microsoft Releases @Ethereum App Development Kit for @Azure #Cloud - @CoinDesk https:/
```

```
head(aws_tweets$text,1)
```

```
## [1] Increasing access to blockchain and ledger databases - #aws #allthingsdistributed  https://t.co/
## 2155 Levels: 'Rackspace have worked with our DevOps engineers to look at our current infrastructure a
```

```
head(google_tweets$text,1)
```

```
## [1] I always worry that Google is actively harming its, say, @googlecloud adoption with all these shu
```

```
## 2170 Levels: 'Cyan'-ara to boring data sets in Sheets! Take a look at how to give your data a fresh
```

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] "Understand the physical structure of Azure infrastructure redundancies and service level agreem
```

```
head(aws_tweets$cleaned_text,1)
```

```
## [1] "Increasing access to blockchain and ledger databases aws allthingsdistributed "
```

```
head(google_tweets$cleaned_text,1)
```

```
## [1] "I always worry that Google is actively harming its say googlecloud adoption with all these shut
```

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

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)
```

```r
#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)
```

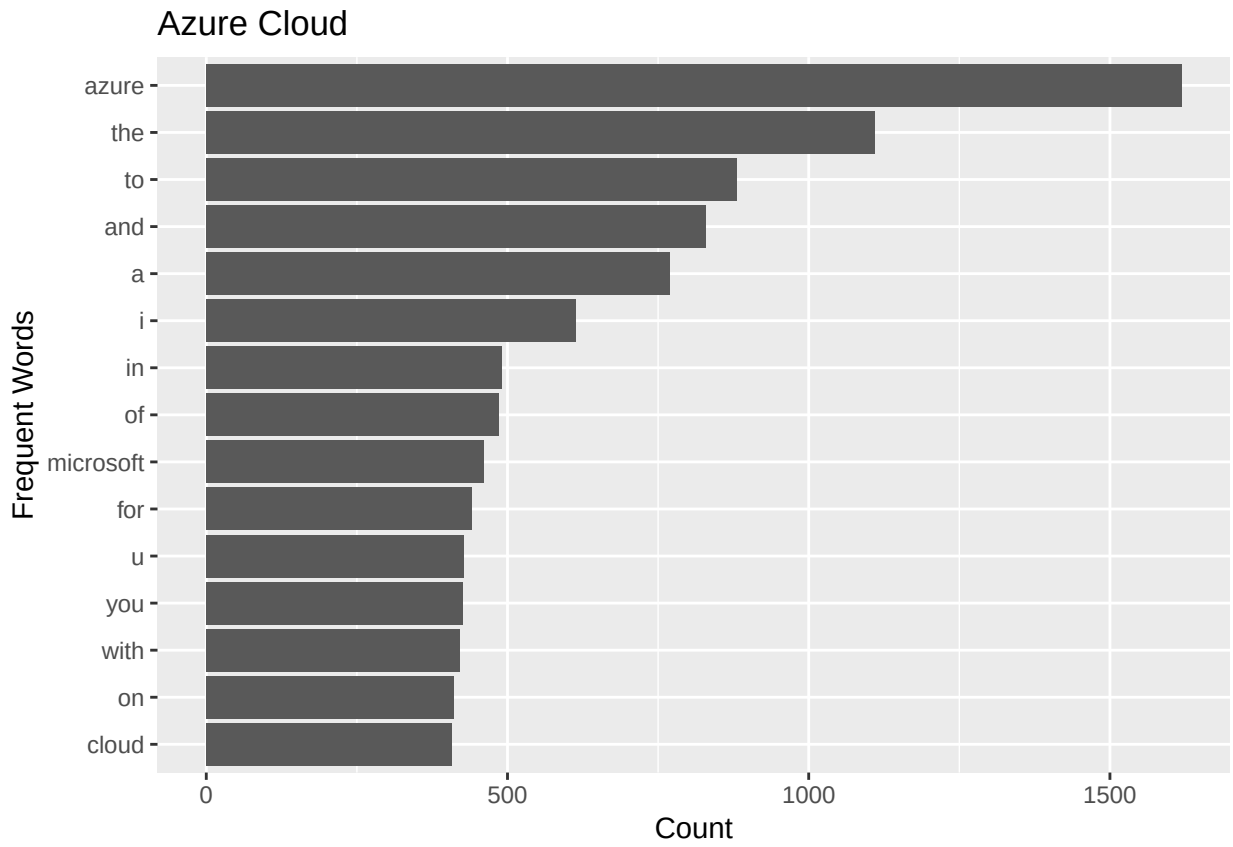
```r
#Split a column into tokens using the tokenizers package
google_tweets_clean <- google_tweets %>%
    dplyr::select(cleaned_text) %>%
    unnest_tokens(word, cleaned_text)

#View the tokenization words
head(google_tweets_clean,2)
```

##Plot the results after cleaning

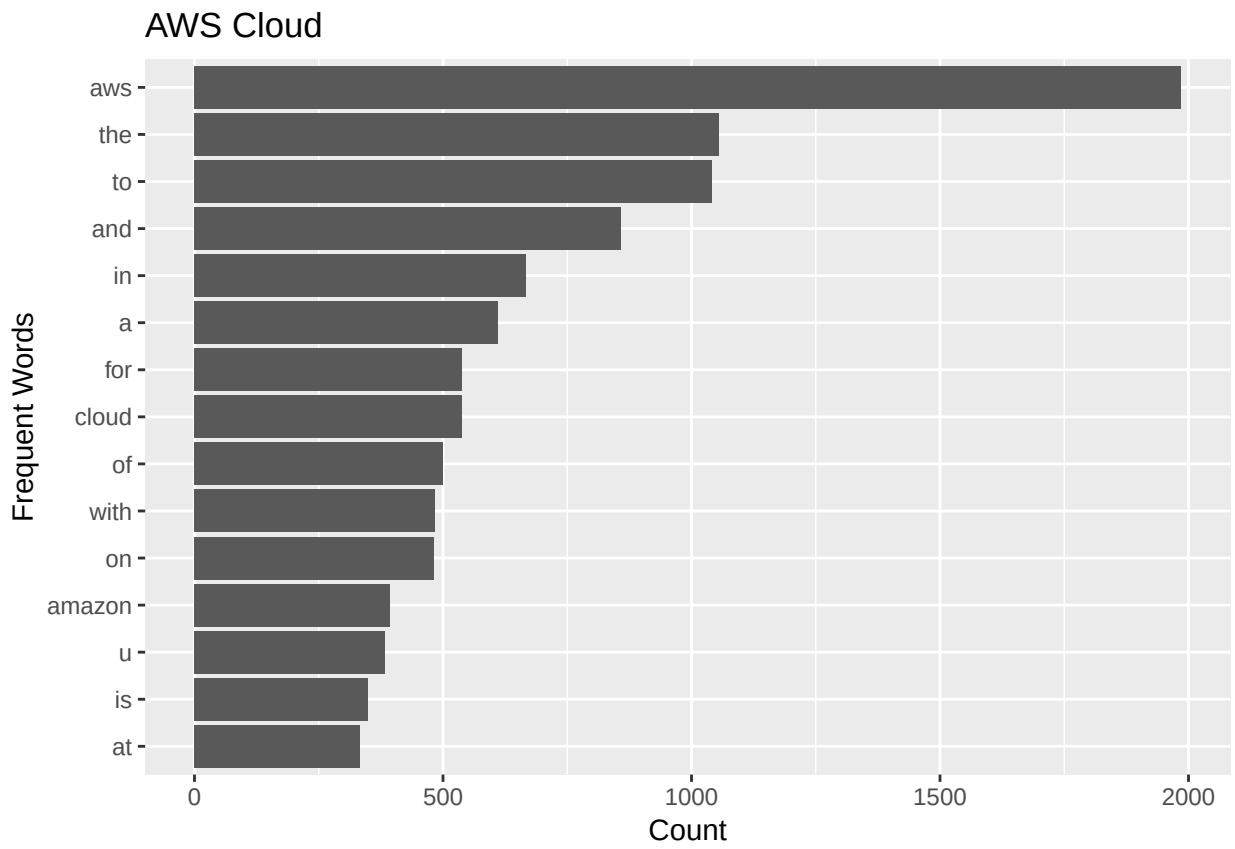
```r
 #Look for any issues by plotting the top 15 common words - Azure
azure_tweets_clean %>%
    count(word, sort = TRUE) %>%
    top_n(15) %>%
    mutate(word = reorder(word, n)) %>%
    ggplot(aes(x = word, y = n))+
    geom_col()+ coord_flip()+
    labs(y = "Count",
         x = "Frequent Words",
         title = "Azure Cloud")
```

```
## Selecting by n
```



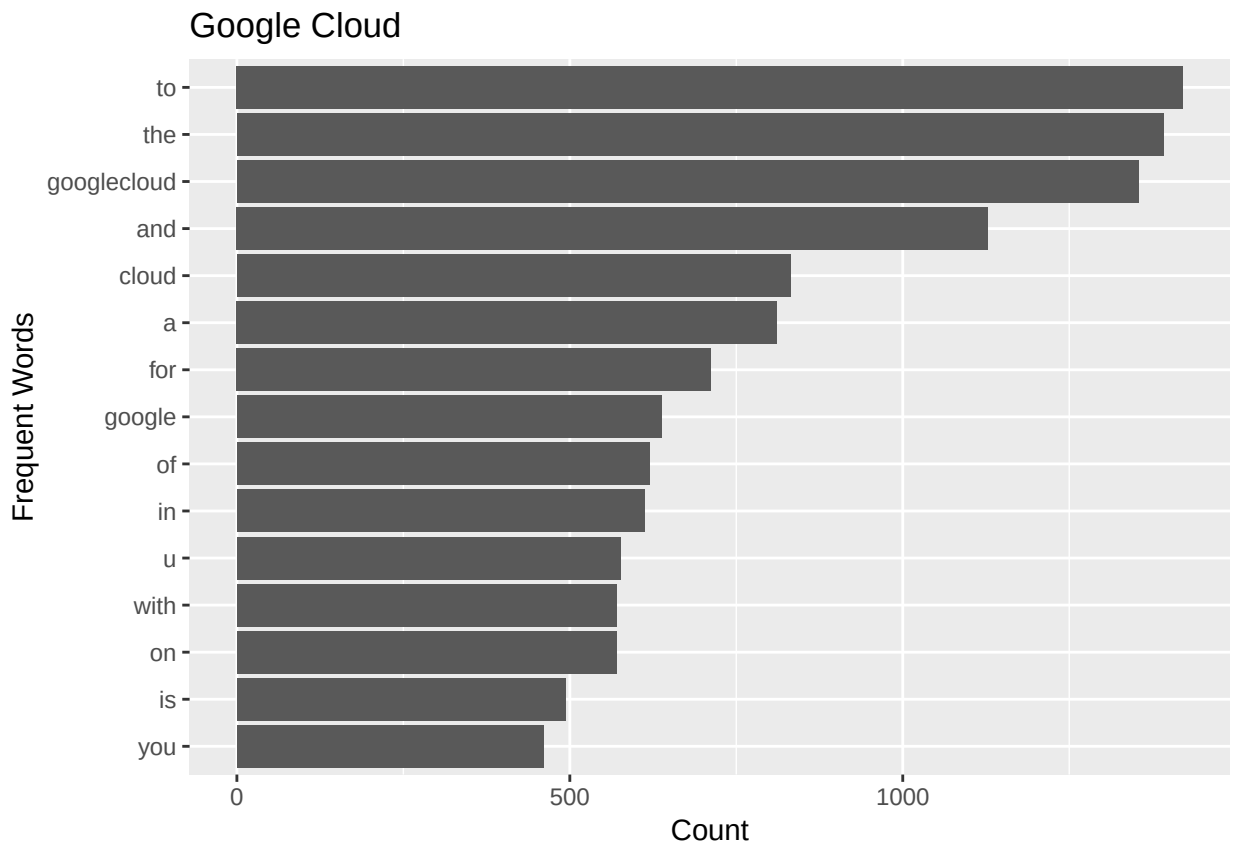

```
#Look for any issues by plotting the most common words - AWS
aws_tweets_clean %>%
    count(word, sort = TRUE) %>%
    top_n(15) %>%
    mutate(word = reorder(word, n)) %>%
    ggplot(aes(x = word, y = n))+
    geom_col()+ coord_flip()+
    labs(y = "Count",
         x = "Frequent Words",
         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)
```

```
#Number of rows BEFORE removing the stop words
nrow(azure_tweets_clean)
```

```
## [1] 40203
```

```
nrow(aws_tweets_clean)
```

```
## [1] 41355
```

```
nrow(google_tweets_clean)
```

```
## [1] 48609
```

```
#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] 21031
```

```
nrow(aws_cleaned_tweet_words)
```

```
## [1] 24091
```

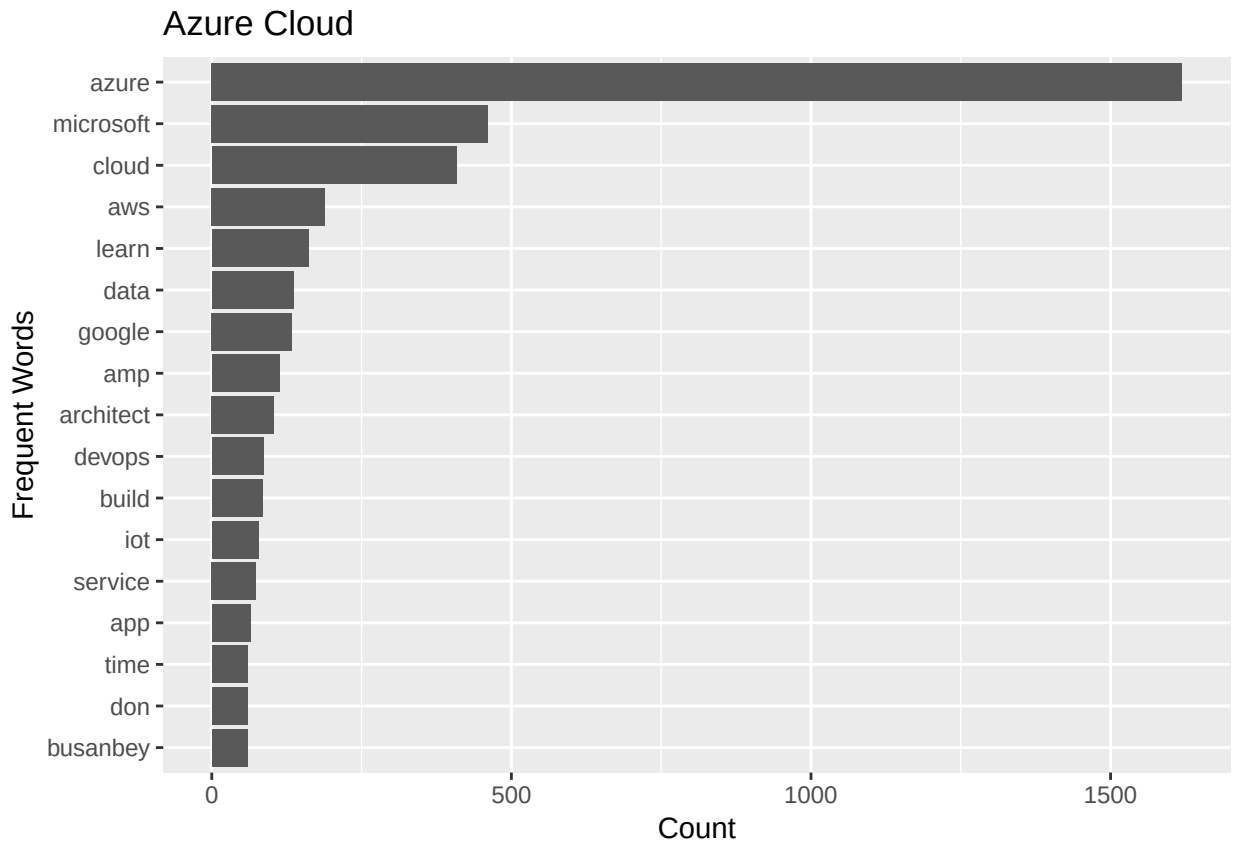
```
nrow(google_cleaned_tweet_words)
```

```
## [1] 25751
```

Plotting for further analysis and check for any remaining issues

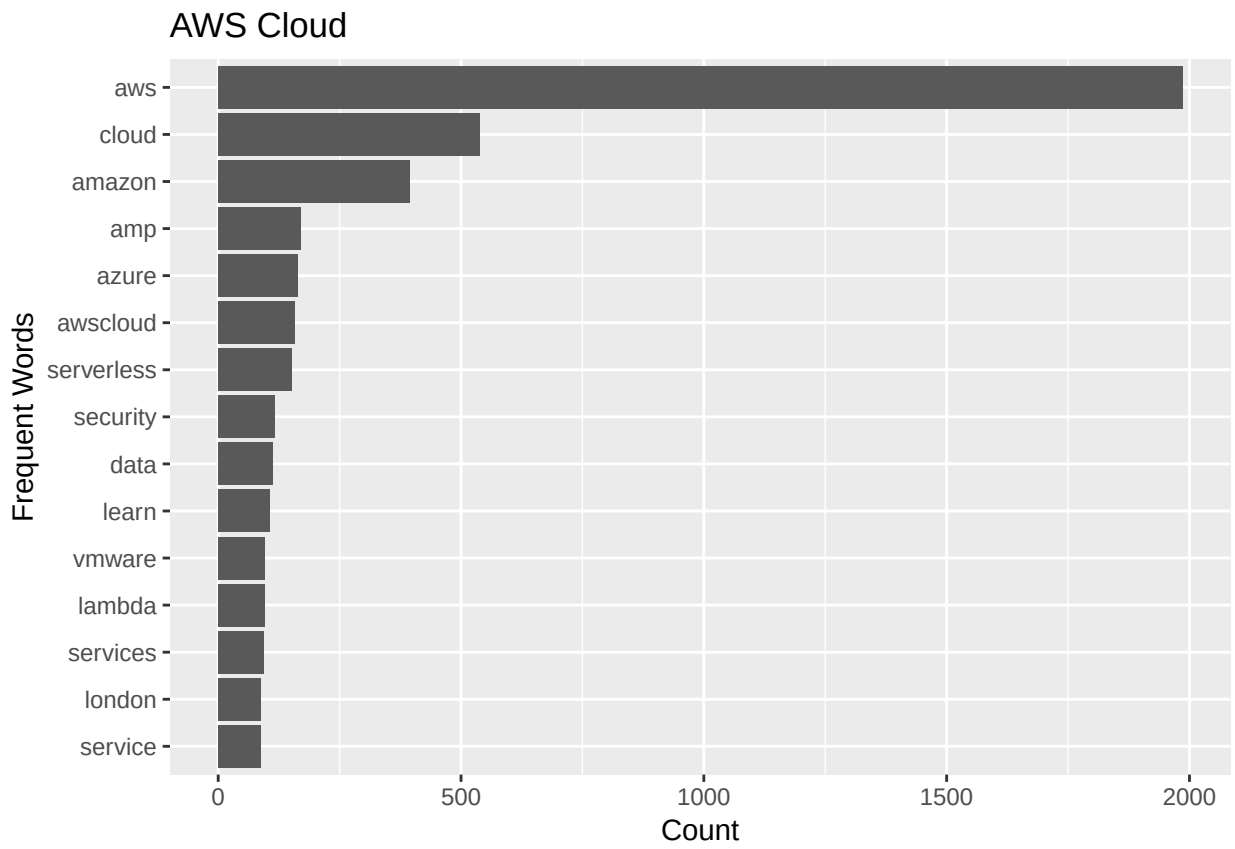
```
# Plot the top 15 words and check if there is any more issues in the data!
azure_cleaned_tweet_words %>%
    count(word, sort = TRUE) %>%
    top_n(15) %>%
    mutate(word = reorder(word, n)) %>%
    ggplot(aes(x = word, y = n)) +
    geom_col()+ coord_flip() +
    labs(y = "Count",
         x = "Frequent Words",
         title = "Azure Cloud ")
```

```
## Selecting by n
```



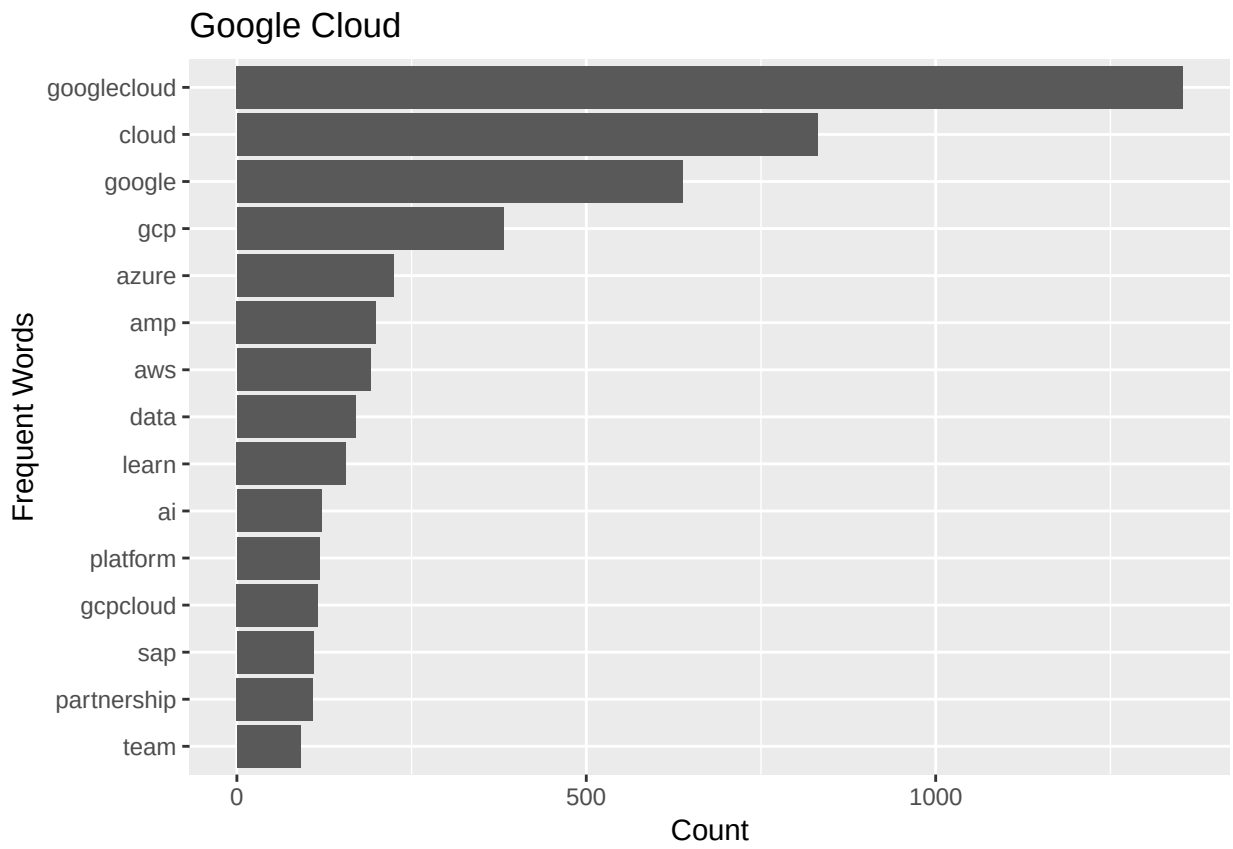

```
aws_cleaned_tweet_words %>%
    count(word, sort = TRUE) %>%
    top_n(15) %>%
    mutate(word = reorder(word, n)) %>%
    ggplot(aes(x = word, y = n)) +
    geom_col()+ coord_flip() +
    labs(y = "Count",
         x = "Frequent Words",
         title = "AWS Cloud ")
```

```
## 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
```



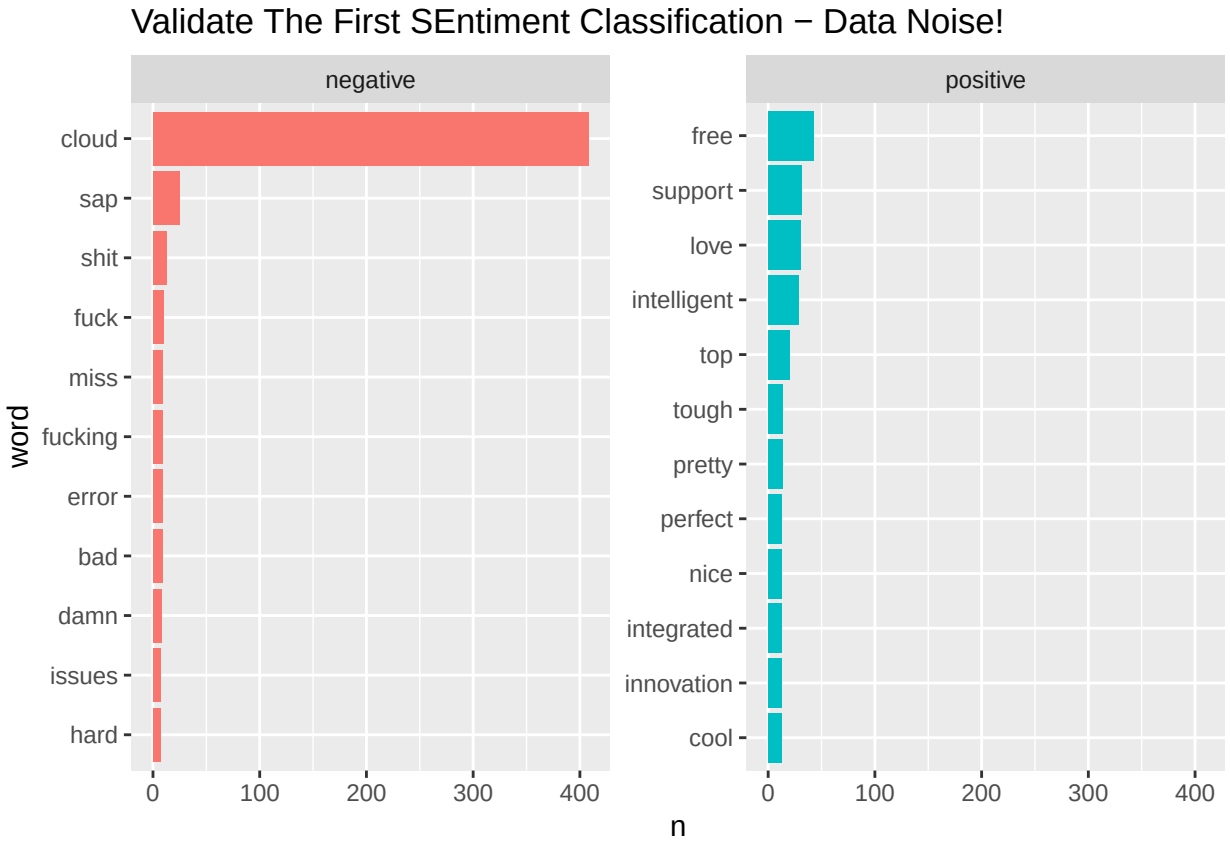
Join with the sentiments from lexicon; either "afinn", "bing", "nrc" and plot the results

```
#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","a
    ),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)
```

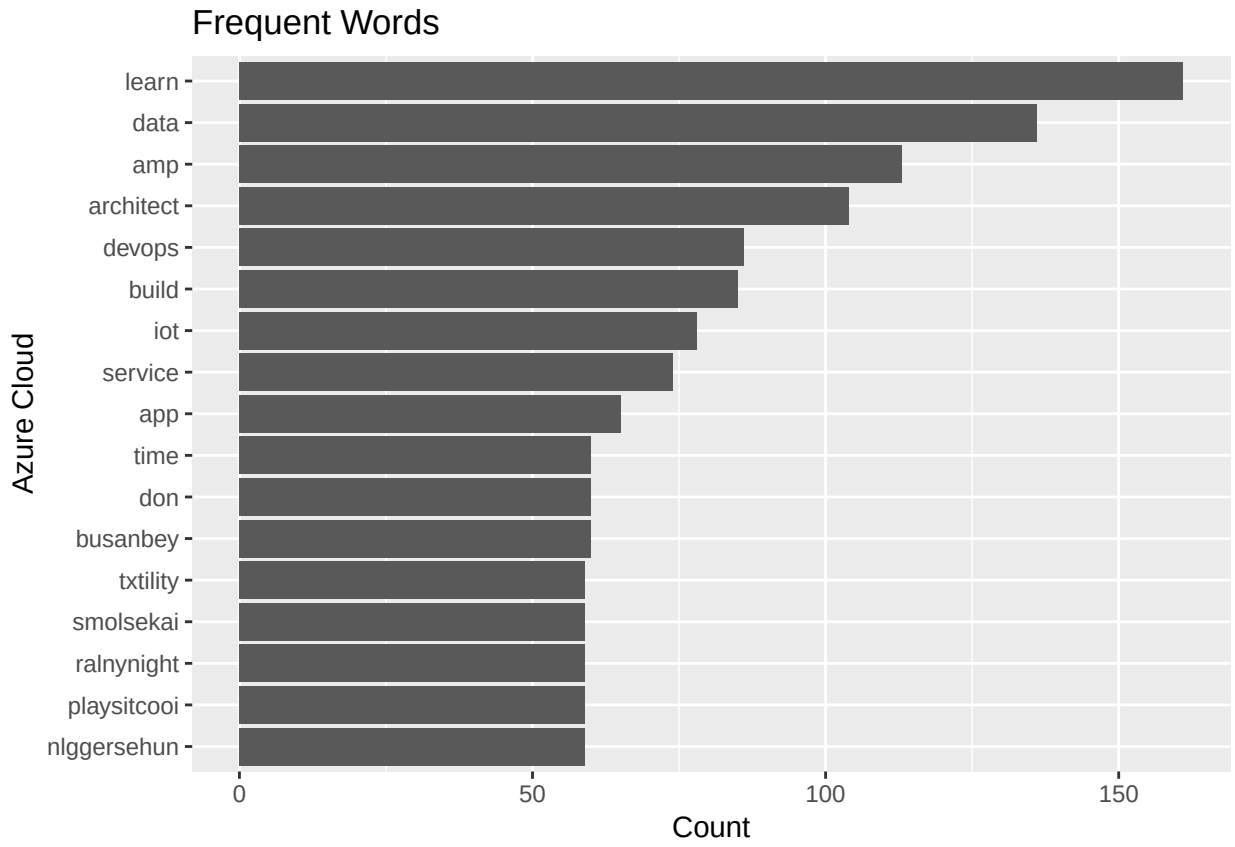
```
## Joining, by = "word"
```

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")
```

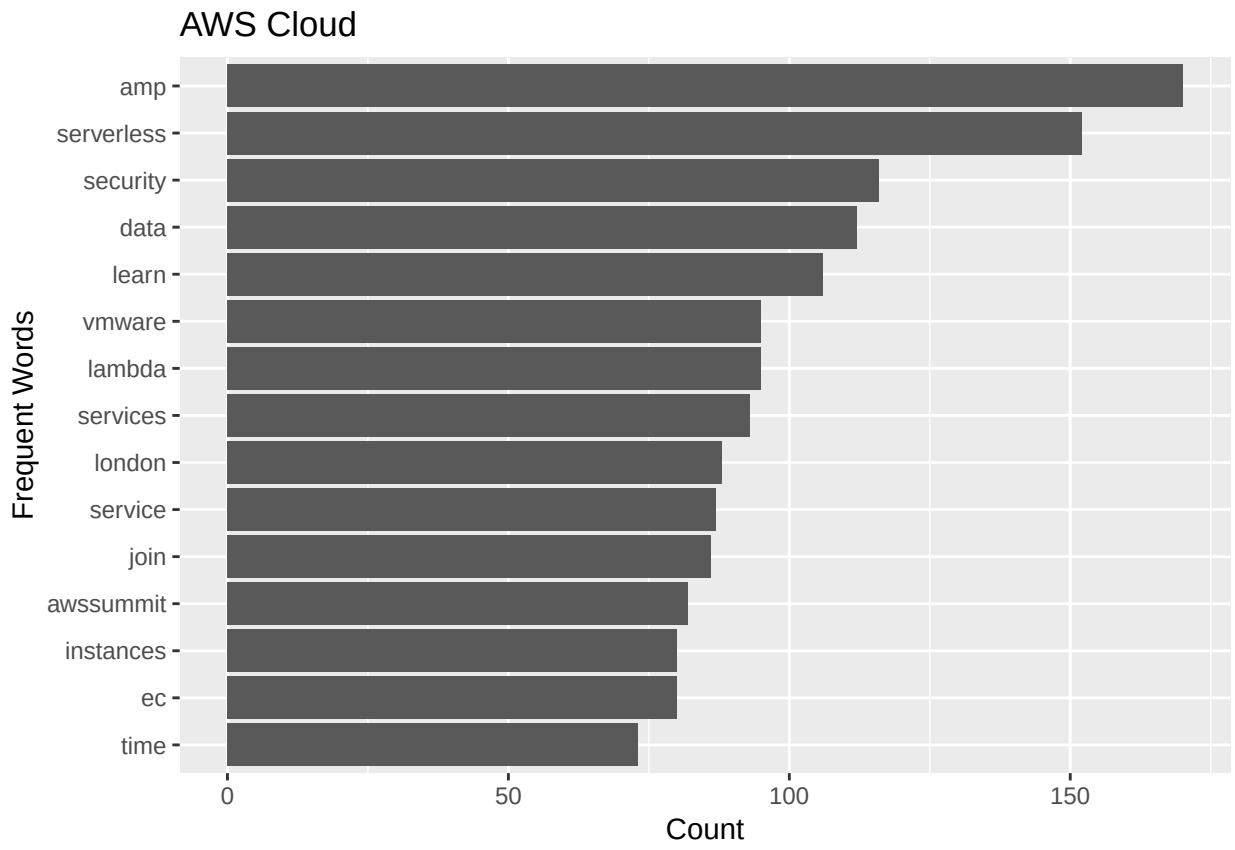
```
## Selecting by n
```




```
#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",
```

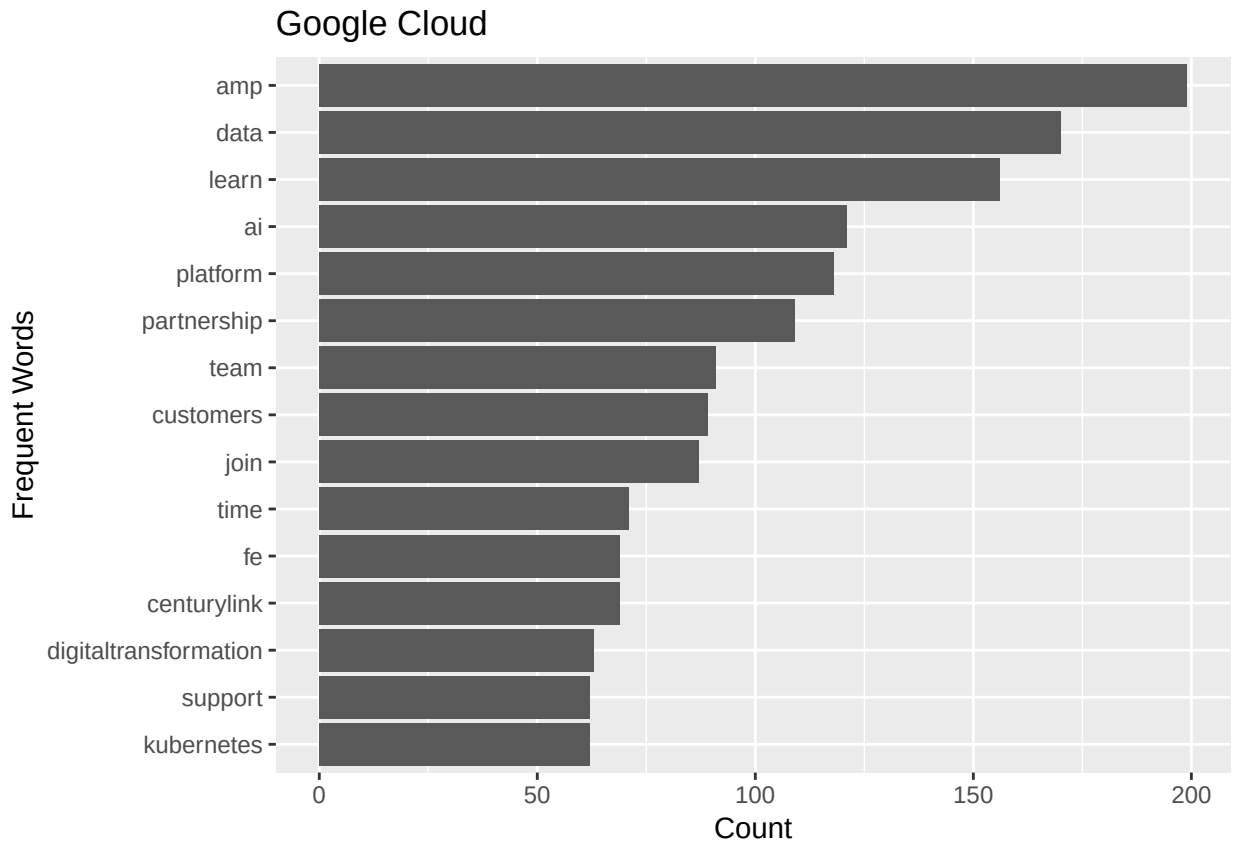
```
         title = "AWS Cloud",
         x = "Frequent Words")
```

```
## 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) %>%
```

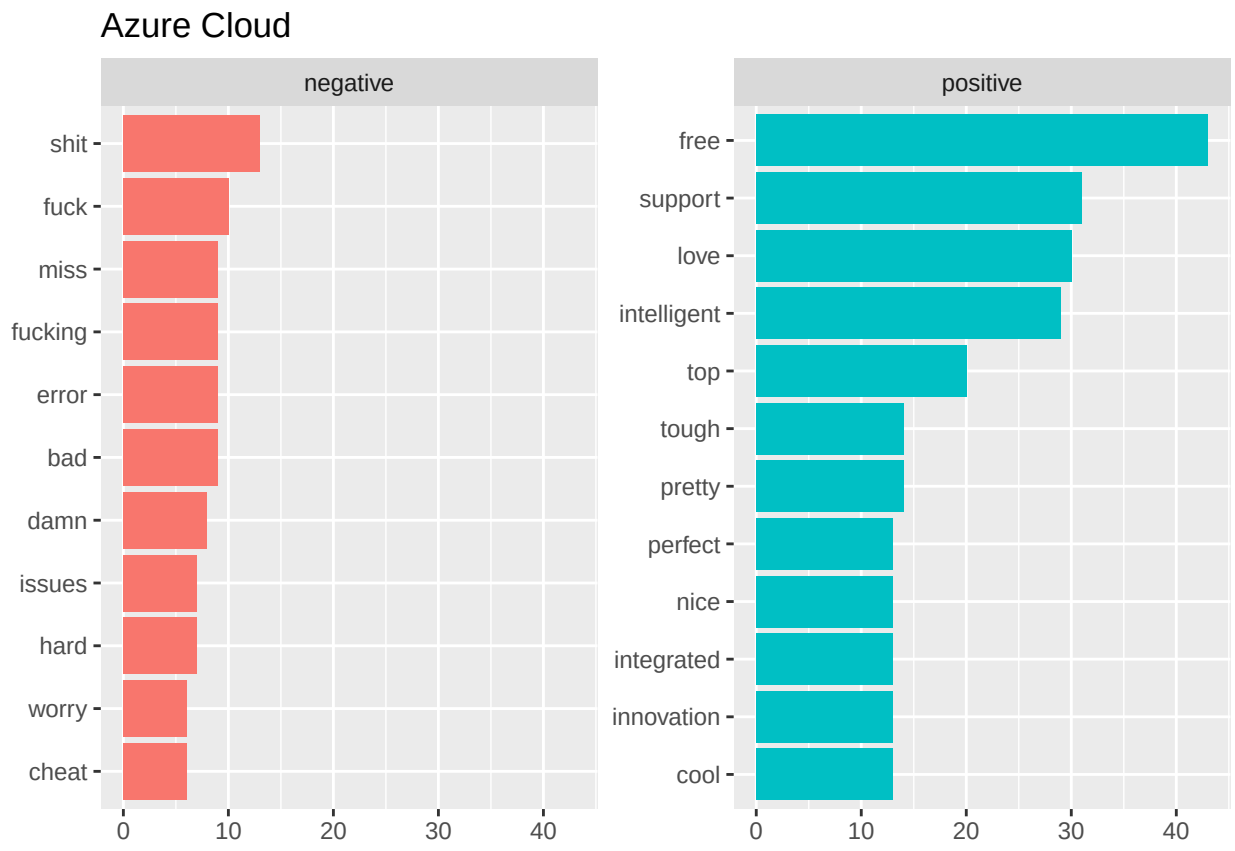
```
    ungroup()
```

```
## 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()
```

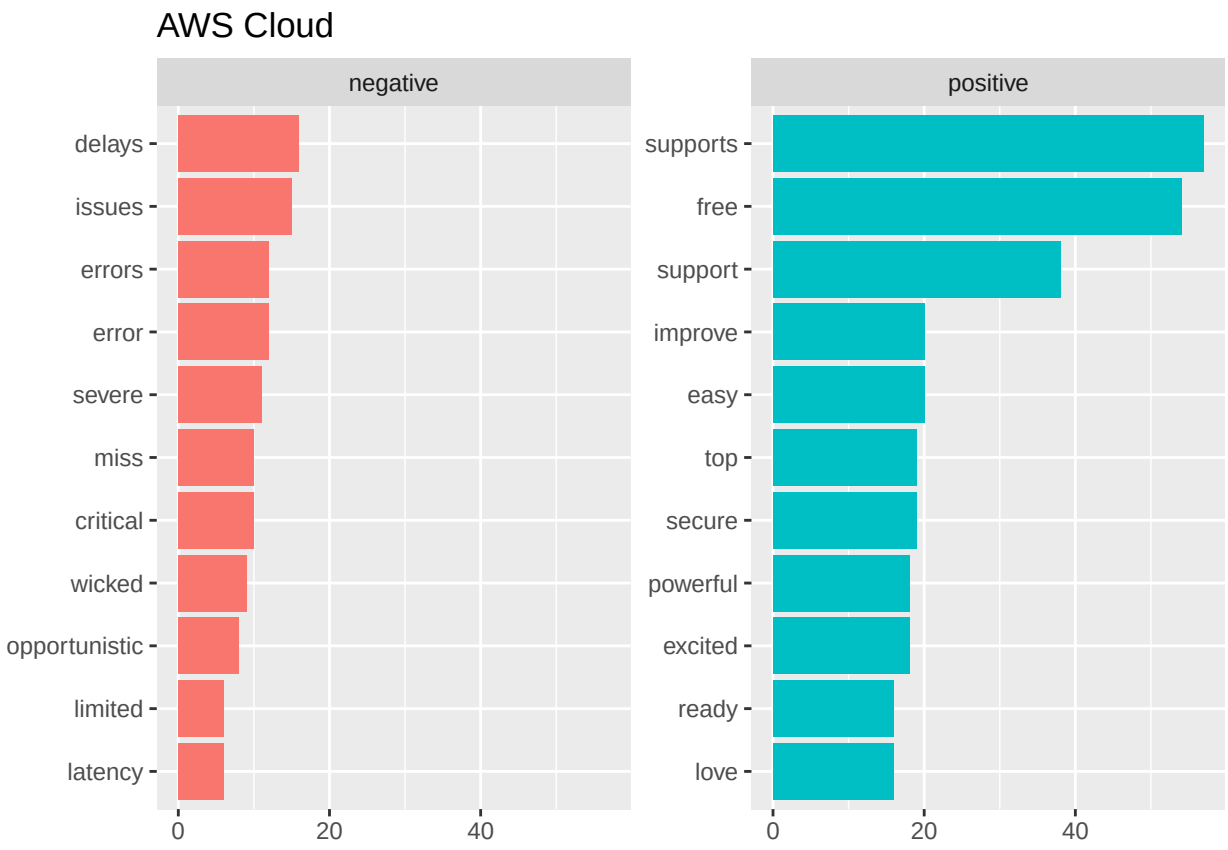
```
## Selecting by n
```




```
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) +
```

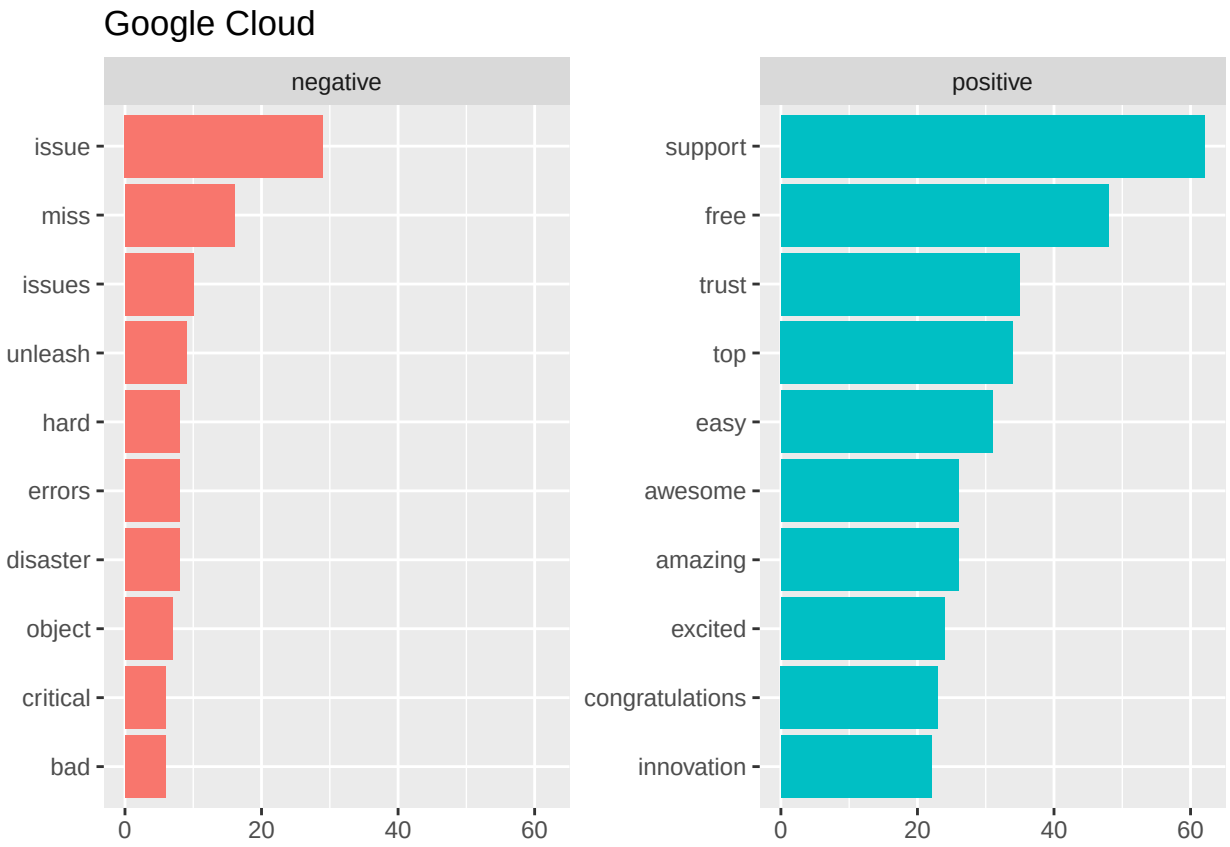
```
    facet_wrap(~sentiment, scales = "free_y") +
    labs(title = "AWS Cloud", x=NULL,y=NULL) + coord_flip()
```

```
## 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

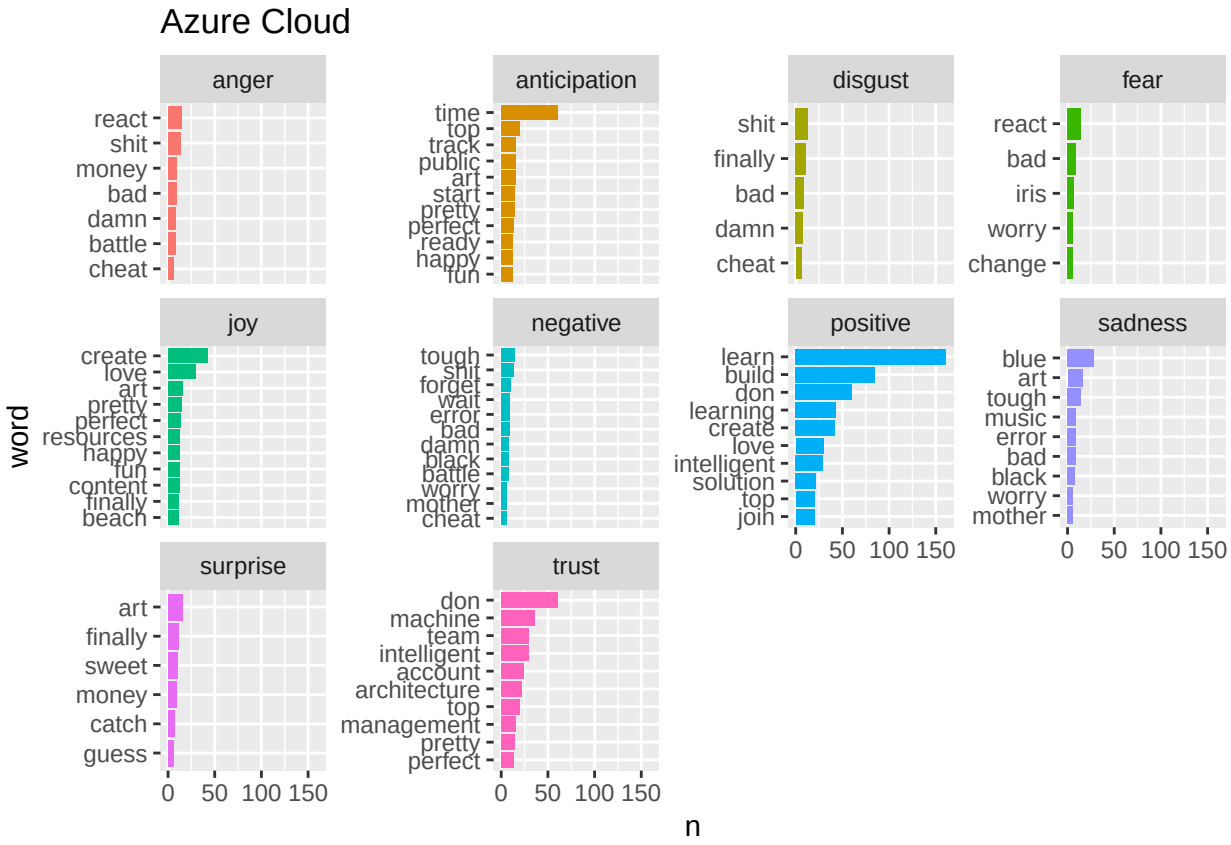
```
#nrc lexicon - Azure
azure_nrc_word_counts <- azure_cleaned_tweet_words %>%
  inner_join(get_sentiments("nrc")) %>%
  count(word, sentiment, sort = TRUE) %>%
  top_n(200)%>%
  ungroup()
```

```
## 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
```



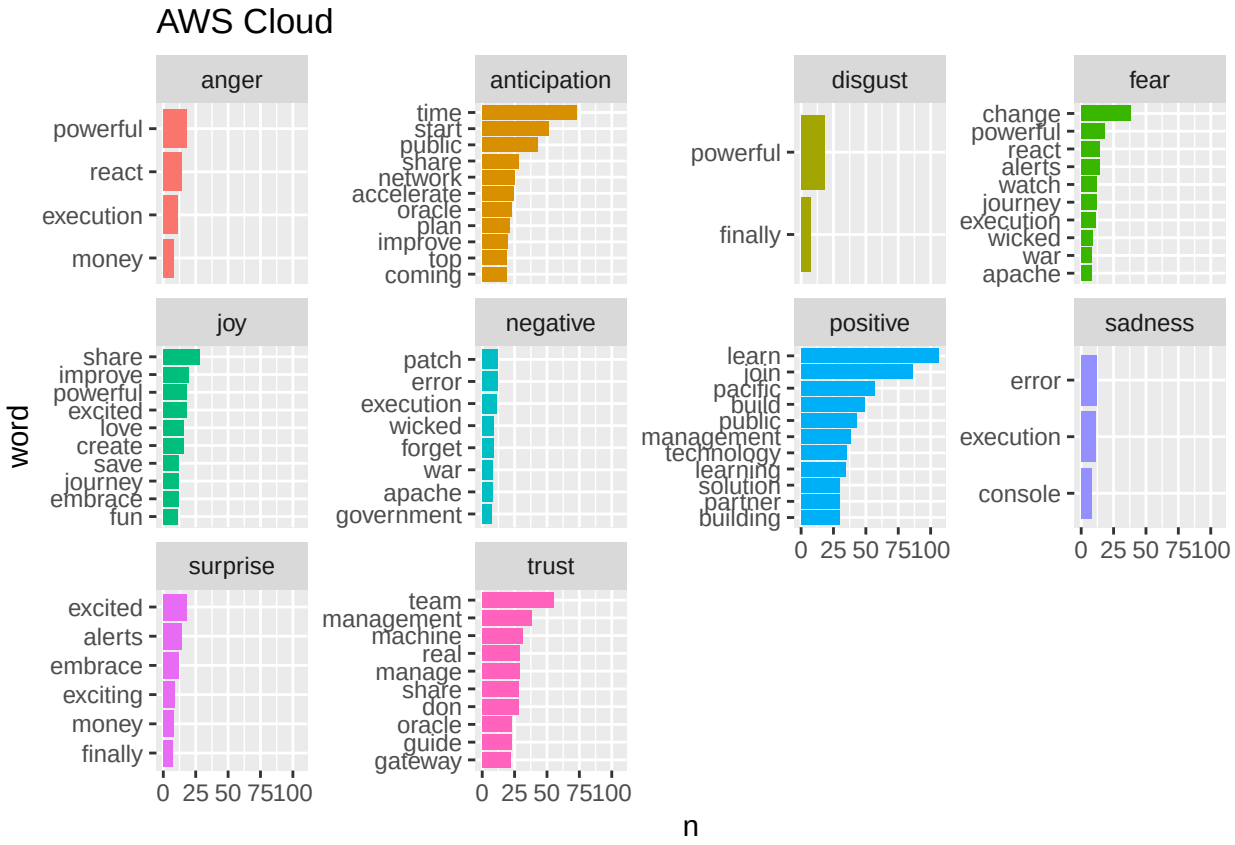

```
#nrc lexicon - AWS
aws_nrc_word_counts <- aws_cleaned_tweet_words %>%
  inner_join(get_sentiments("nrc")) %>%
  count(word, sentiment, sort = TRUE) %>%
  top_n(200)%>%
  ungroup()
```

```
## 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
```



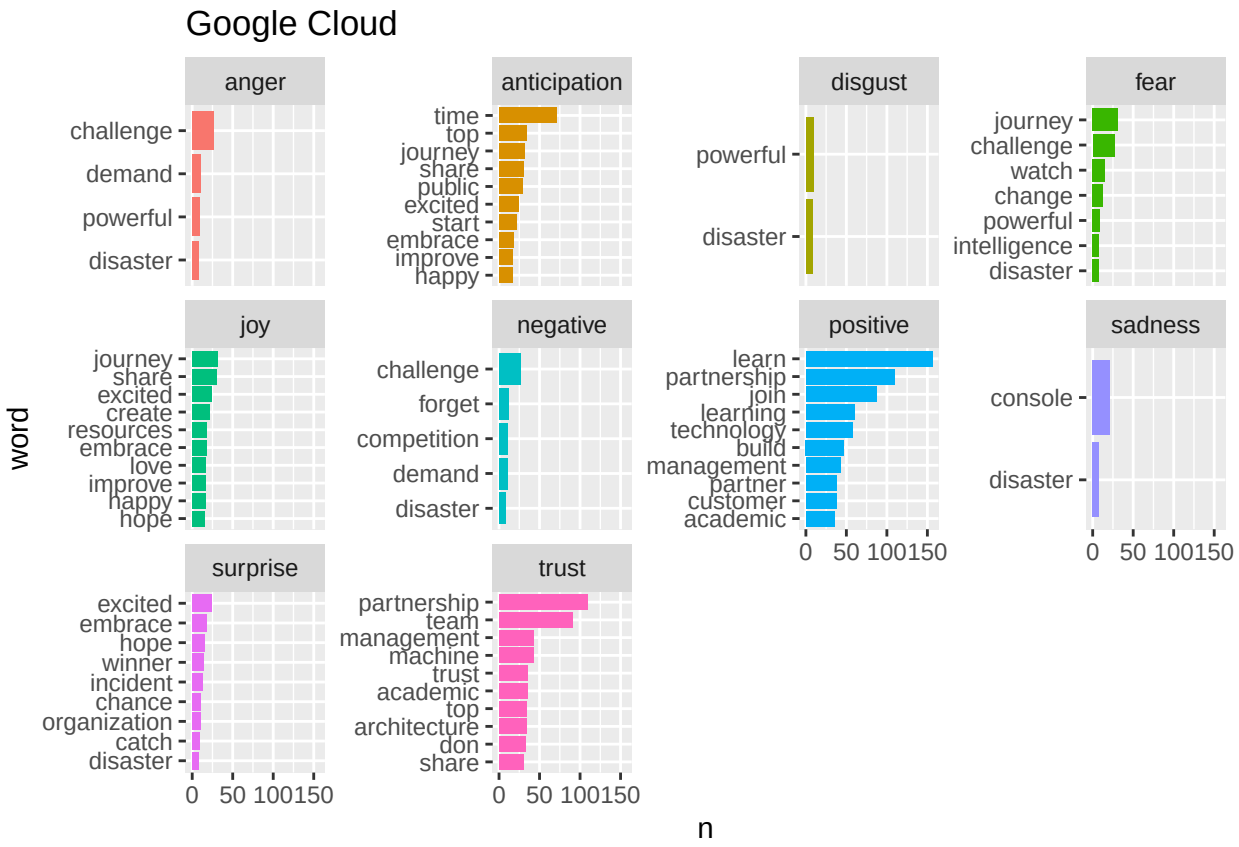

```
#nrc lexicon - Google
google_nrc_word_counts <- google_cleaned_tweet_words %>%
  inner_join(get_sentiments("nrc")) %>%
  count(word, sentiment, sort = TRUE) %>%
  top_n(200)%>%
  ungroup()
```

```
## 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()
```

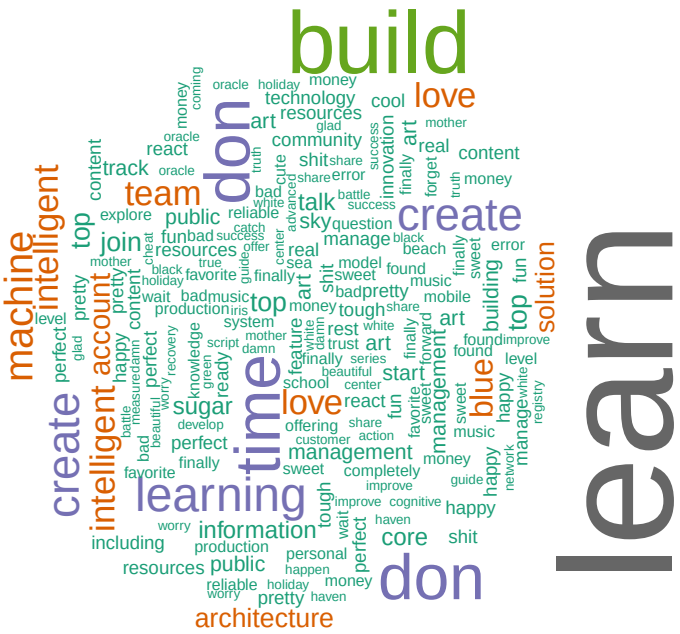
```
## Selecting by n
```



Cloudwords

```
#Cloudwords PACKAGE
#Todo: invistigate a bug in cloudword library. when previos plots in memory, the plots breaks therefore

#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 )
```


```r
#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 )
```

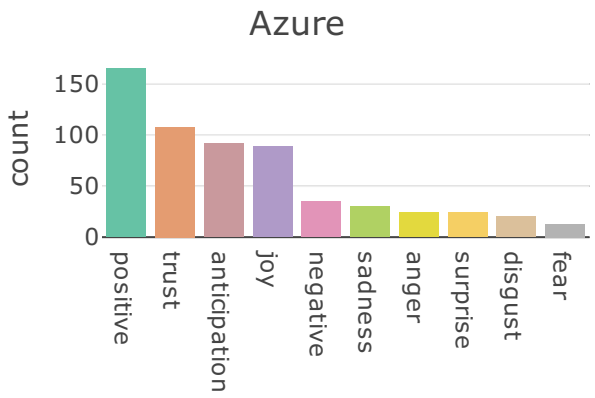
Get the sentiment score for each emotion - nrc

```
# 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
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
```

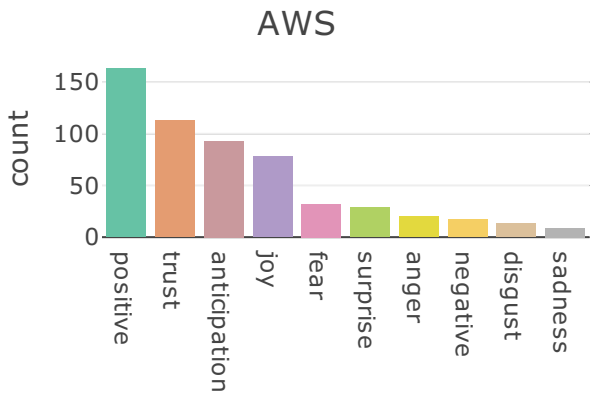
```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```




```
#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_s
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
```

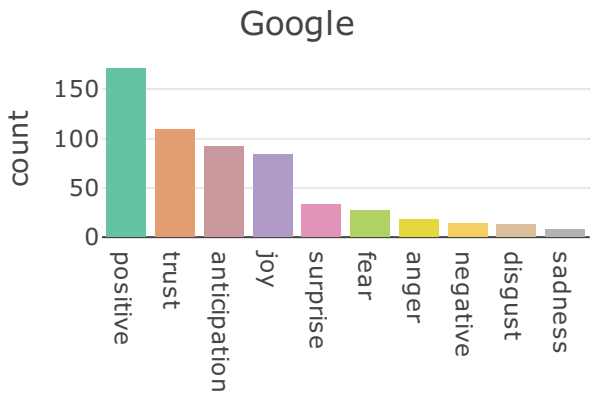
```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```




```
#Google
google_sentiment<-get_nrc_sentiment(as.vector(google_nrc_word_counts$word))
google_sentiment_df<- data.frame(count=colSums(google_sentiment), emotion=names(colSums(google_sentimen
google_sentiment_df$emotion = factor(google_sentiment_df$emotion, levels=google_sentiment_df$emotion[or
plot_ly(google_sentiment_df, x=~emotion, y=~count, type="bar", color=~emotion) %>%
  layout(xaxis=list(title=""), showlegend=FALSE, title="Google")
```

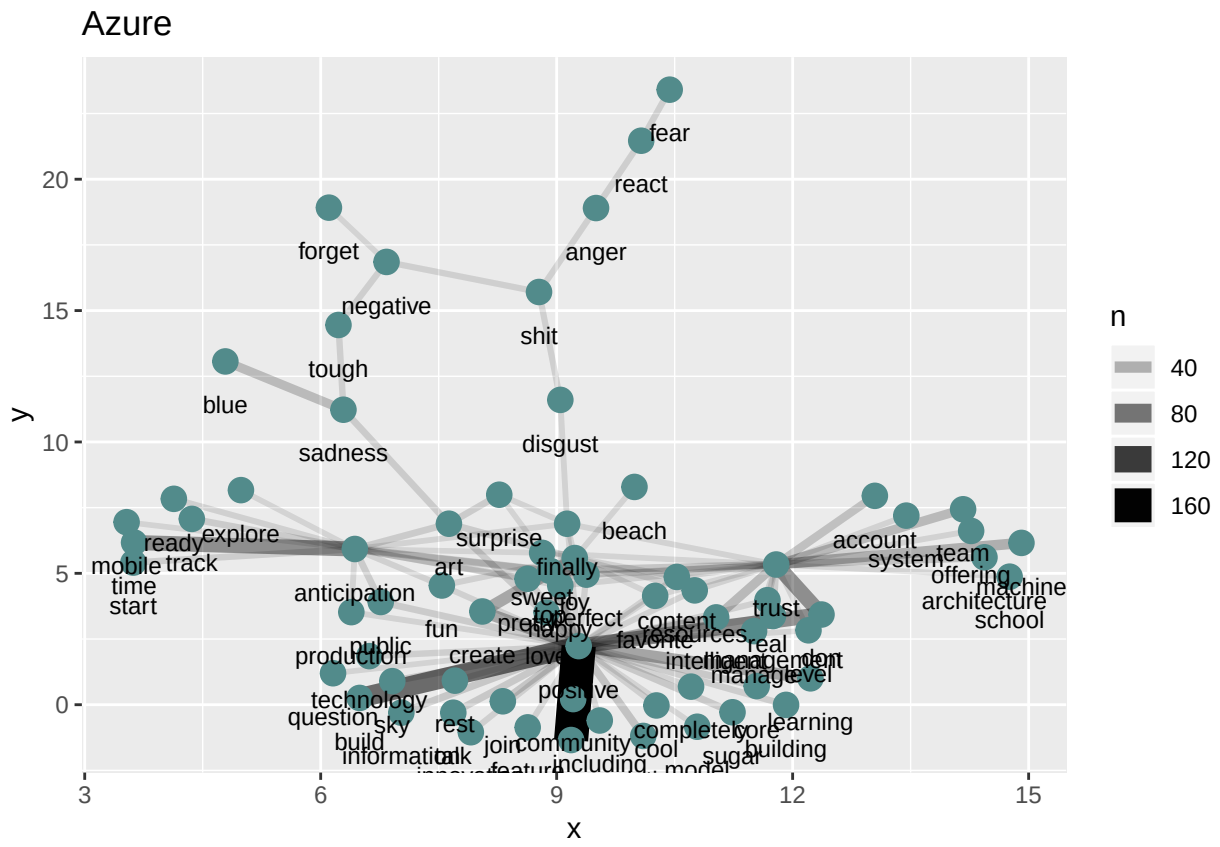
```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors

## 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
```

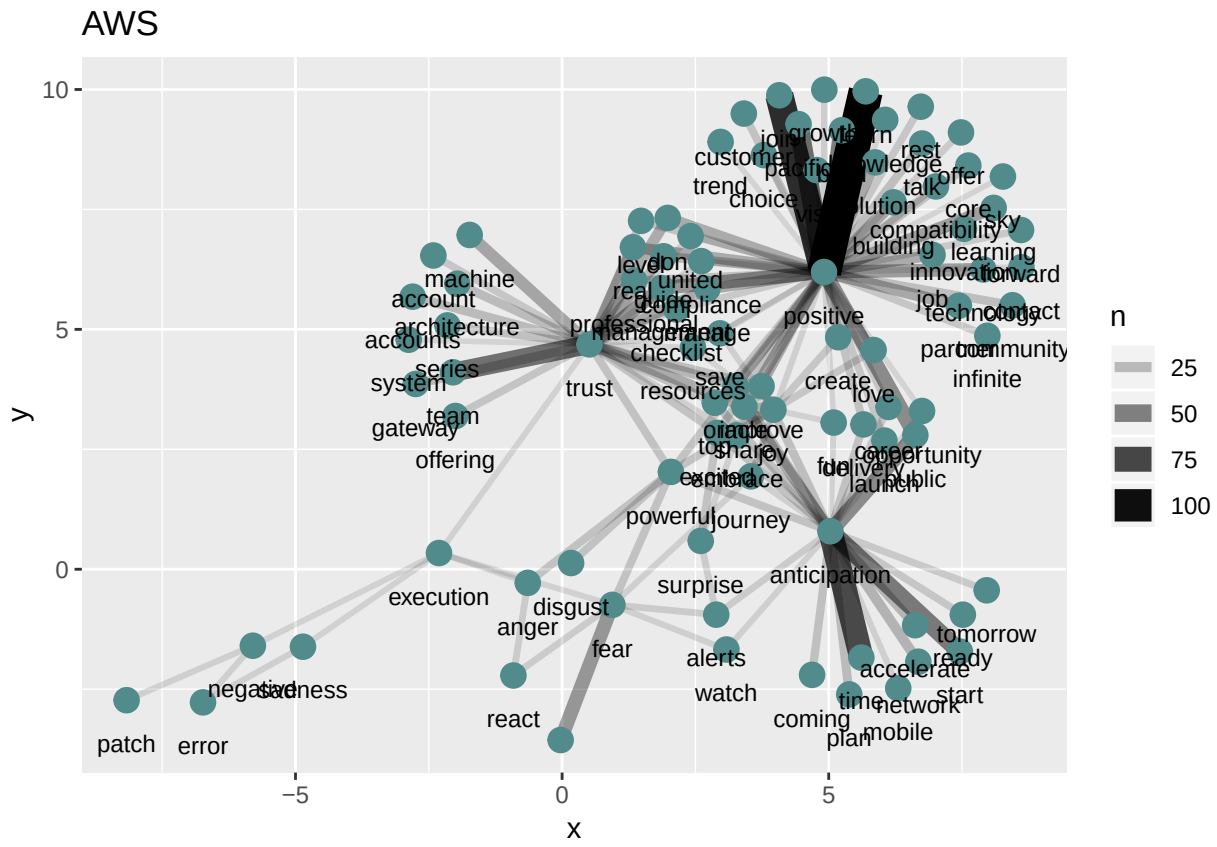


Plot word network with minimum frequency of 10

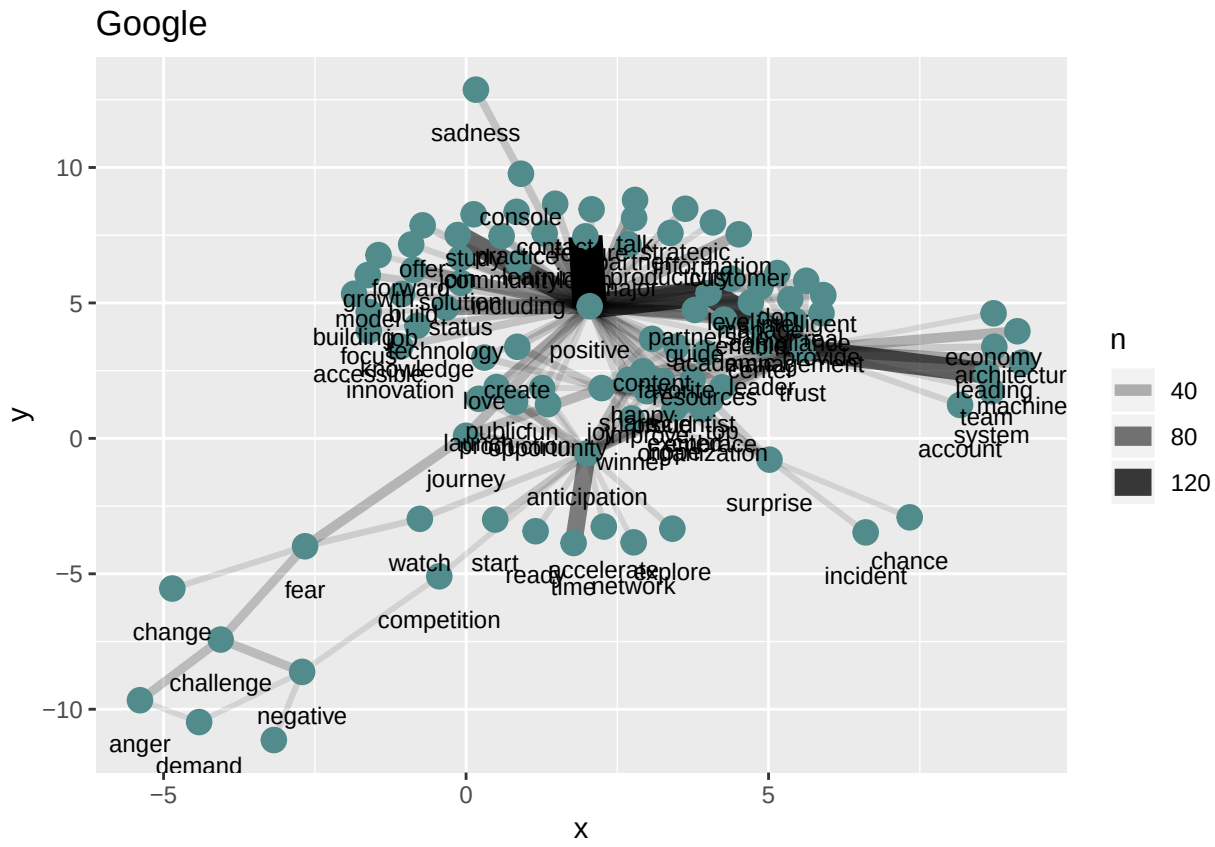
```
#plot word network with minimum frequency of 10
#Azure
azure_nrc_word_counts %>%
    filter(n >= 10) %>%
    graph_from_data_frame() %>%
    ggraph(layout = "fr") +
    geom_edge_link(aes(edge_alpha = n, edge_width = n)) +
    geom_node_point(color = "darkslategray4", size = 4) +
    geom_node_text(aes(label = 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")
```




```r
#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")
```


Google
sadness
console
talk
strategic
information
customer
offer
forward
growth
solution
including
model
build
status
building
job
technology
positive
focus
knowledge
accessible
innovation
create
love
content
leader
resources
trust
economy
architectur
leading
machine
team
system
account
public
fun
journey
anticipation
surprise
chance
incident
watch
start
ready
accelerate
explore
time
network
fear
competition
change
challenge
negative
anger
demand
winner
n
40
80
120
x
y
10
5
0
−5
−10
−5
0
5

Thank You