Final Project

Neil Farrugia 17336831

14/11/2021

Part 1: Analysis

The dataset that was chosen for this first of the analysis is on wine.
It can be found on kaggle using this link: https://www.kaggle.com/natashasavc/ie-students-winemag-analysis/ We will be analysing the relationship between prices and quality of wines and also we will be looking at what makes a great wine.

Reading in the data

```
#data can be found here https://www.kaggle.com/natashasavc/ie-students-winemag-analysis

#ie. year and price/points
#Packages needed for this section
library(ggplot2)


#read in the data
df <- read.csv("winemag-data-130k-v2.csv")
nas <- apply(apply(df, 2, is.na),2,sum)
#Quick visualization plot to find missing values
barplot(nas,main="Missing values investigation",
        ylab="num of NAs",las=2)
```


```
#what are the dimensions
on <- dim(df)[1]
vn <- dim(df)[2]
sum <- c("n observation"=on, "n variables" = vn)
print(sum)
```

```
## n observation   n variables
##        129971            14
```

From this brief missing values exploration, we can see that there is only 1 column that contains missing values - points column. With just over 8000 missing values.

The total number of observation is 129971 and the total number of variables is 14.

There is a high number of observation in this data set. Completely omitting rows with missing Points value will still allow plenty of data points to draw conclusions from.

```
#removing all NA values
df1 <- df[complete.cases(df),]
```

Analysis on distrubition of prices and points

The Data contains some interesting variables, two will be focused on for this next section, which are the Prices and Points of the wine.

It is important to note that points were accredited by a random professional taster, which carries no bias when scoring the wines.

Data manipulation prior analysis

```
#print Summary of prices
summary(df1$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4.00   17.00   25.00   35.36   42.00 3300.00
```

A brief a exploration of the data was preformed prior to running the code below. Since the 3rd Quartile is 42 and the max is 3300. It seems that most wines are on the cheaper side and only very few are very expensive

This big range, might skew our analysis, thus removing the outliers might be the best option.

```
#Hypothesis - Very few wines are above the $200/300 mark
#These next few lines - finds the range of bottles at certain price that covers
#99.5% of the data set, this number was chosen through trial and error
tp <- table(df1$price)
p99 <- ceiling(length(df1$price)*0.995)
rngp99 <-(p99-20):(p99+20) #Give or take $20

#cumulative sum function - to count up wines that are priced from 0th percentile
#to the 99.5th percentile
tpRS <- cumsum(tp)
pless <- tp[which(tpRS%in%rngp99)]
plessLabel <- labels(pless[length(pless)])

mostexp <- max(df1$price)

print(paste0("$1 to $",plessLabel," covers 99.5% of the total dataset"))
```

```
## [1] "$1 to $224 covers 99.5% of the total dataset"
```

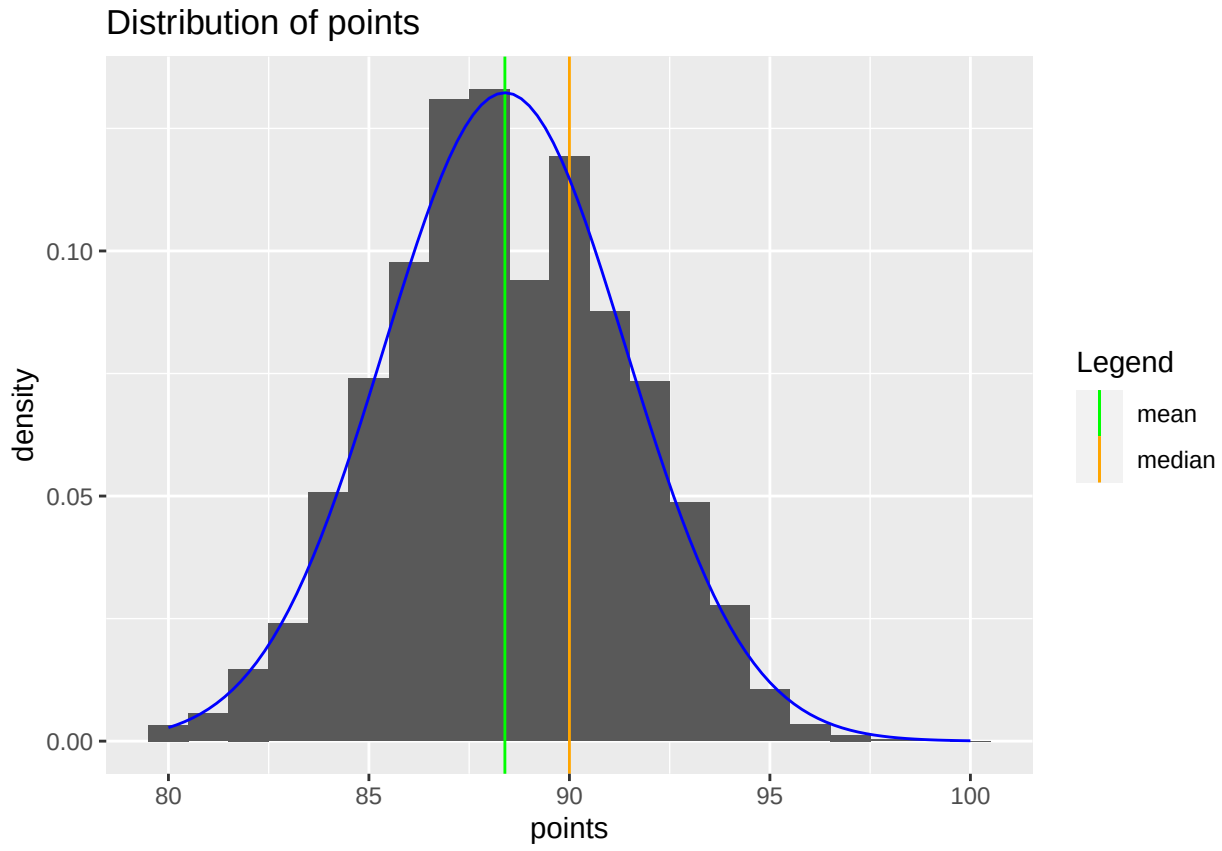
The calculations show a big range of prices, with under 99.5% of the wines below $225 and the most expensive wine is 3300 dollars.
Removing these outliers will allow for a better a visualizations of the analysis and distributions.
A better visualization of the data analysis and distribution can be achieved by removing outliers, so any wine with a price point less than 225 dollars or greater than 3300 dollars will be removed.

```
#revoming any wines which prices are above 225$ (99.5 percentile)
df1 <- df1[which(df1$price < as.numeric(plessLabel)),]
```

Analyis on prices and points of wines

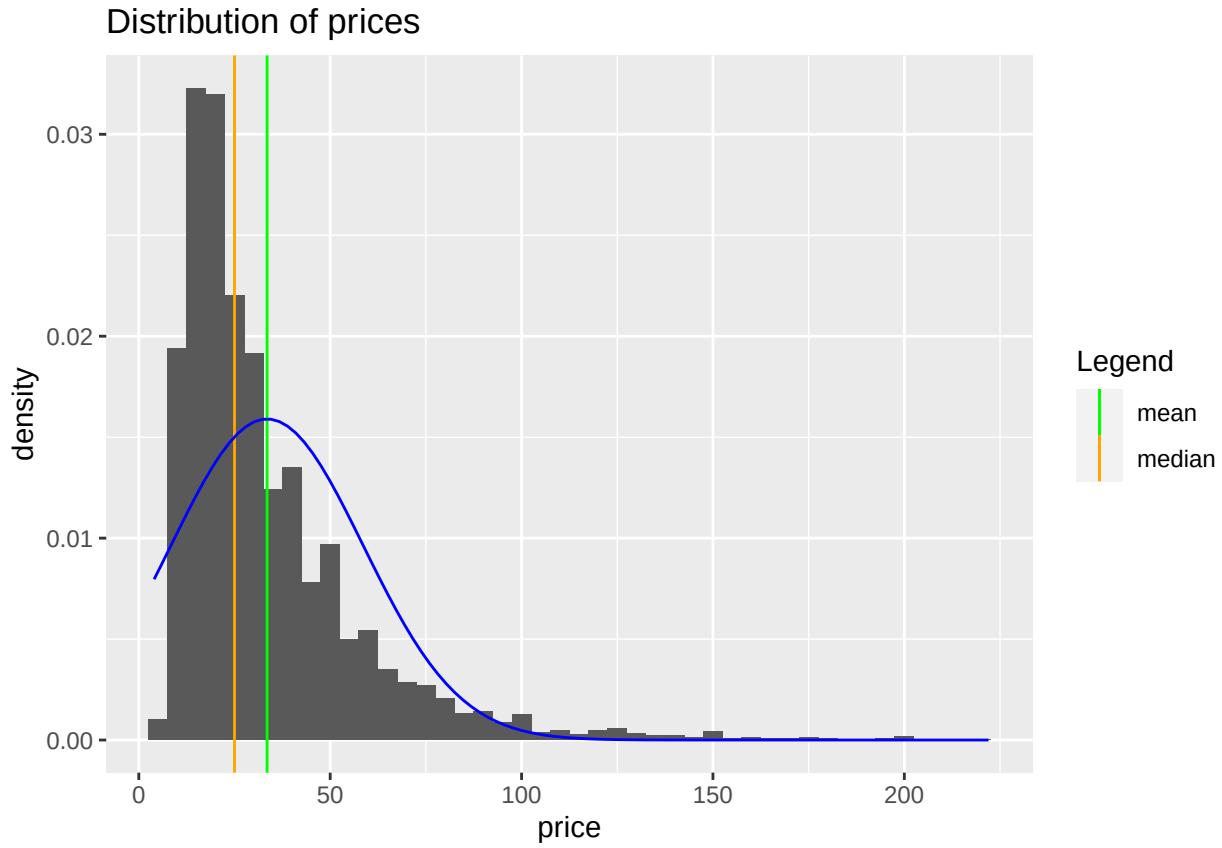
Into the analysis part.
A distribution of prices and points will be plotted.

```r
#Points and price analysis
avg <- mean(df1$points)
median <- median(80:100)

#Colors for the graph
colors <- c("mean" = "green", "median" = "orange")

#grpahing distribution of points
ggplot(df1,aes(x=points))+
  geom_histogram(binwidth = 1,aes(y=..density..))+
  #geom_vline(aes(xintercept = mode,color="mode"))+
  geom_vline(aes(xintercept = avg,color="mean"))+
  geom_vline(aes(xintercept = median,color="median"))+
  labs(title="Distribution of points",
         color = "Legend") +
 stat_function(fun=dnorm, args = list(mean = mean(df1$points), sd= sd(df1$points)), col = 'blue')+
 scale_color_manual(values = colors)
```




```r
#p dont need this
#Summary of mean and median of price
avgP <- mean(df1$price)
#modeP <- mode(df1$price)
medianP <- median(df1$price)
sumP <- c("mean Price" = avgP, "median Price" = medianP)
```

```
#p dont neeed this
#look at wine prices, for different countries
#only look at top 10 countries with most wines in this dataset
ctab <- sort(table(df1$country), decreasing = T)
ctab10 <- labels(ctab[1:10])[[1]]
df10 <- df1[which(df1$country%in%ctab10),]


#Distribution of prices in the dataset
ggplot(df1,aes(x=price))+
  geom_histogram(binwidth = 5,aes(y=..density..))+
  #geom_vline(aes(xintercept = mode,color="mode"))+
  geom_vline(aes(xintercept = avgP,color="mean"))+
  geom_vline(aes(xintercept = medianP,color="median"))+
  labs(title="Distribution of prices",
         color = "Legend") +
 stat_function(fun=dnorm, args = list(mean = mean(df1$price), sd= sd(df1$price)), col = 'blue')+
 scale_color_manual(values = colors)
```



It is interesting to note the distribution of prices and points. They are both quite different from each other. As you can see the points follow a normal distribution, with 80 being the "worst" wine and 100 being the "best" wine. Where as the prices the distribution is highly skewed to the right, with most wines hovering around the mean and the median (35.36 and 25 respectively).
Analysis on the price of wines in the top 11 countries:

```r
#How many countries?
length(unique(df1$country))
```

```
## [1] 43
```

There are a total of 43 countries. For ease of visualization, only the eleven countries that have the most observations in the dataset will be used in this price analysis.

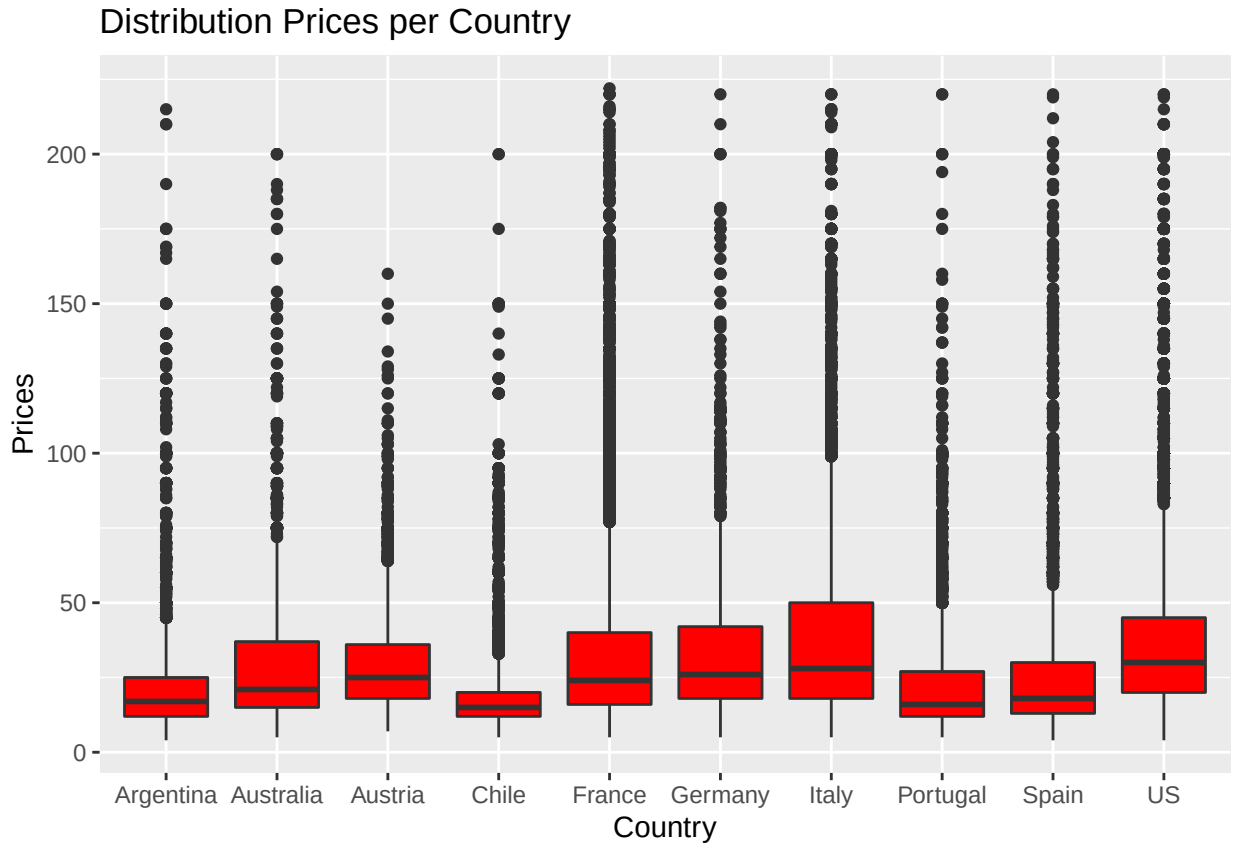
```r
#Selecting only the 11 countries with the most wines in the dataset
topCon <- labels(sort(table(df1$country),decreasing = T)[1:11])[[1]]
df102 <- df10[which(df10$country %in% topCon),]
topCon
```

```
##  [1] "US"          "France"      "Italy"       "Spain"       "Portugal"
##  [6] "Chile"       "Argentina"   "Austria"     "Australia"   "Germany"
## [11] "New Zealand"
```

These top 11 countries are also among st the top wine producing countries according to this website, https://worldpopulationreview.com/country-rankings/wine-producing-countries/

Plotting the price distrbutions for the countries

```r
#why do I need this function
avgPvC<-aggregate(df102$price,by=list(country=df102$country),FUN=mean)

#box plot for the prices in different countries
ggplot(df102,aes(x = country,y=price))+
  geom_boxplot(fill = "red")+
  labs(title="Distribution Prices per Country",
         y = "Prices", x = "Country")
```



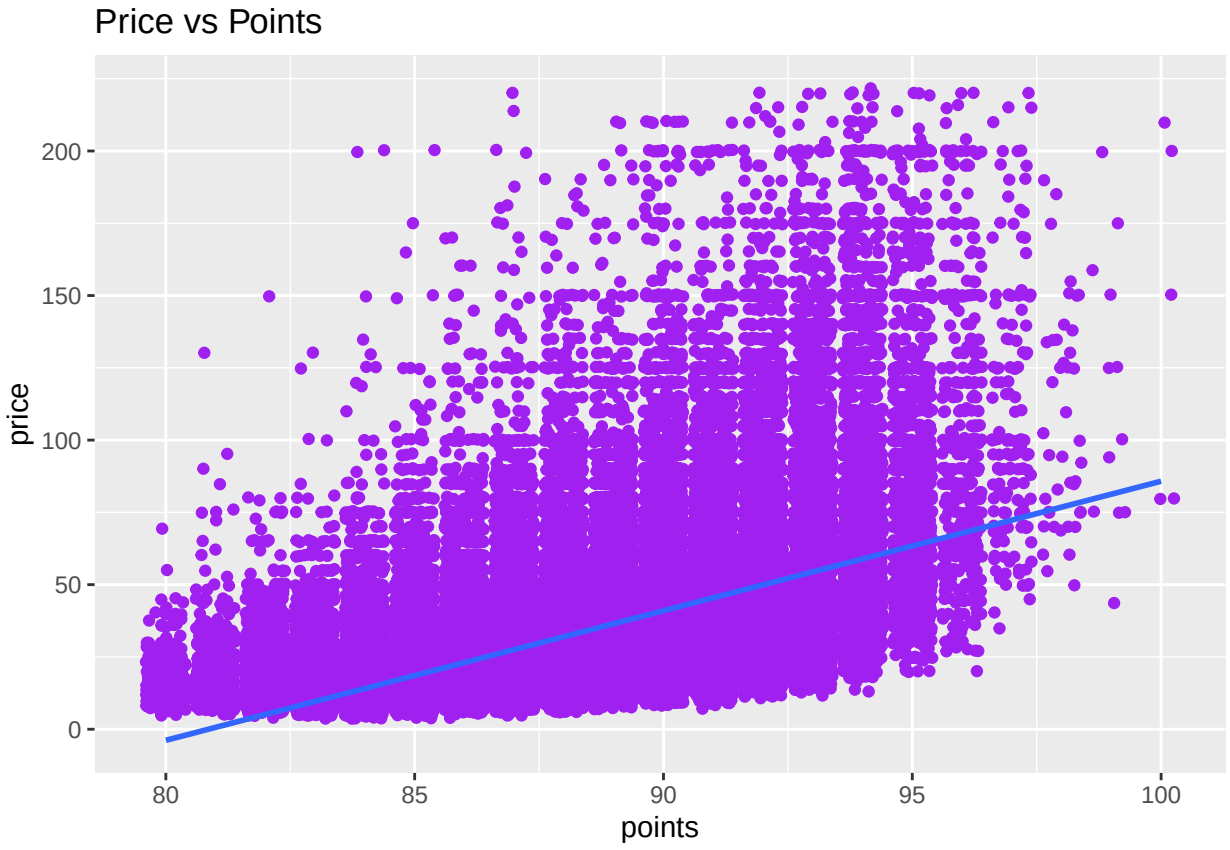
The conclusions that can be drawn from these boxplot are:
1. US,Italy,France,Germany and Italy all seem to produce more expensive wines in comparison to the others.
2. Chile has the least amount of variance and also produces the cheapest wines with Argentina not far off.
3. Italy has the biggest range of prices.
4. Spain and Portugal are very similar

Analysis on the correlation between price and points

In order to get a better understanding of the relationship between prices and quality of wine, a correlation graph will plotted with a regression line.

```
# correlation graph
#Because points is ordinal data -
#"jitter" is used to in order to better visualize the trend
ggplot(df10,aes(x=points,y=price))+
  geom_point(position="jitter",color = "purple") +
  geom_smooth(method="lm", se=F)+
  labs(title = "Price vs Points")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
rs <- summary(lm(price~points,data=df10))$r.squared
print(paste("The coefficient of determination =",round(rs,2)))
```

```
## [1] "The coefficient of determination = 0.29"
```

We can see that there is definitely an uptrend from the graph, although it isn't very strong. This is also confirmed by the coefficient of determination which can be summarised as positive with weak/medium strength.
Intuitively this makes sense because there should be a degree of correlation between price and wine quality. Just like every products in life quality and price are linked. Yet the relation isn't overly strong meaning there must other factors at play that may contribute, for example, to great tasting cheaper wines or bad tasting expensive wines. Maybe a country like Chile, which has cheaper wines, must also produce great quality wines at a cheaper price in comparison to the US or France.

Text analysis on wine descriptions

This dataset also provided a wine description for each wine. I have studied wine and know that powerful adjectives are used to describe wines that are designed to appeal to our sense of taste and smell. Hence, the analysis of the specific words used could be interesting.
For this analysis 3 packages will be used, they have not been covered in the course.
Tokenizers is a package that converts text data into tokens. Which is usually the first step to text processing and natural language processing as well.
The process of tokenizing is essential for the analysis of the wine's descriptive data. It allows the process

of counting up the ocurrences of each of word. Moreover this package, it allows you to provide a stopword list. This is a list of words that you want to exclude in the process of tokenization. The stopword package, therefore, provides a list of most frequent words in the english language that are usually of no benefit to analysis like these ones. Words for example, "the" and "and".
Ggworld packages works hand in hand with ggplot. It helps produce a wordcloud, a useful way to visualize word frequency data. It plots words, and the size of each word is propotional to the frequency of occurences.

```
#https://cran.r-project.org/web/packages/tokenizers/vignettes/introduction-to-tokenizers.html

library(stopwords)
library(tokenizers)
library(ggwordcloud)
citation("tokenizers")
```

```
##
## To cite the tokenizers package in publications, please cite the paper
## in the Journal of Open Source Software:
##
##   Lincoln A. Mullen et al., "Fast, Consistent Tokenization of Natural
##   Language Text," Journal of Open Source Software 3, no. 23 (2018):
##   655, https://doi.org/10.21105/joss.00655.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {Fast, Consistent Tokenization of Natural Language Text},
##     author = {Lincoln A. Mullen and Kenneth Benoit and Os Keyes and Dmitry Selivanov and Jeffrey Arn
##     journal = {Journal of Open Source Software},
##     year = {2018},
##     volume = {3},
##     issue = {23},
##     pages = {655},
##     url = {https://doi.org/10.21105/joss.00655},
##     doi = {10.21105/joss.00655},
##   }
```

```
citation("ggwordcloud")
```

```
##
## To cite package 'ggwordcloud' in publications use:
##
##   Erwan Le Pennec and Kamil Slowikowski (2019). ggwordcloud: A Word
##   Cloud Geom for 'ggplot2'. R package version 0.5.0.
##   https://CRAN.R-project.org/package=ggwordcloud
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {ggwordcloud: A Word Cloud Geom for 'ggplot2'},
##     author = {Erwan {Le Pennec} and Kamil Slowikowski},
##     year = {2019},
```

```
##     note = {R package version 0.5.0},
##     url = {https://CRAN.R-project.org/package=ggwordcloud},
##   }
```

```
citation("stopwords")
```

```
##
## To cite package 'stopwords' in publications use:
##
##   Kenneth Benoit, David Muhr and Kohei Watanabe (2021). stopwords:
##   Multilingual Stopword Lists. R package version 2.3.
##   https://CRAN.R-project.org/package=stopwords
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {stopwords: Multilingual Stopword Lists},
##     author = {Kenneth Benoit and David Muhr and Kohei Watanabe},
##     year = {2021},
##     note = {R package version 2.3},
##     url = {https://CRAN.R-project.org/package=stopwords},
##   }
```

Word cloud plot of most frequent words

```
#isolate the description of each wine
rev <- df1$description

#The list obviousWords was update regularly when a word was found that was not covered in the stopwword
obviousWords <- c("wine","flavors","now","nose","well","show","like","aroma","drink","body","offer","vi

#Also included any region/name of wines.
#Only interested in descriptive wine words
stopwords = c(stopwords::stopwords("en"),tolower(df1$variety),df1$province,df1$country,obviousWords)

#The unlist function was used on the tockenized object as we are interested in the whole dataset not ju
#The function table or count does not work on list
#wds <- unlist(tokenize_ngrams(rev,n=2,n_min=1,stopwords=stopwords))
wds <- unlist(tokenize_word_stems(rev,stopwords=stopwords))#note will take long to run
wdsdf <- data.frame(sort(table(wds),decreasing=T))

#plotting a word cloud
ggplot(wdsdf[1:100,],
       aes(label=wds,color = factor(sample.int(10, 100, replace = TRUE)),size=Freq))+
  geom_text_wordcloud() +
  labs(title = "Word Cloud for dataset")+
  theme_minimal()
```

Word Cloud for dataset

100 of the most frequent words were plotted for better graphical aesthetics.
The most frequent words are fruit, palate, aroma, acid etc.

This provides a nice overview of the token dataset, however it probably isn't precise enough to draw accurate conclusions.

Worst wines vs Best wines description

Next to take this one step further, we will be grouping the data into two groups, the best wines and the worst wines (relative to this dataset).
And then analyzing the words used to describe these two groups. This time using a bar chart, instead of a word cloud.
It will be interesting to investigate whether different words are used for the groups. Whether you can tell if a wine scored highly solely by looking at the description of the wine.

```
#This package allows the graph to be put side by side, for ease comparison
library(gridExtra)

#After trial an error using the best wines (points >96) and worst wines (points<81), provided a big eno
dfG <- df1[which(df1$points>96),]
dfB <- df1[which(df1$points<81),]

#Same process as before
```

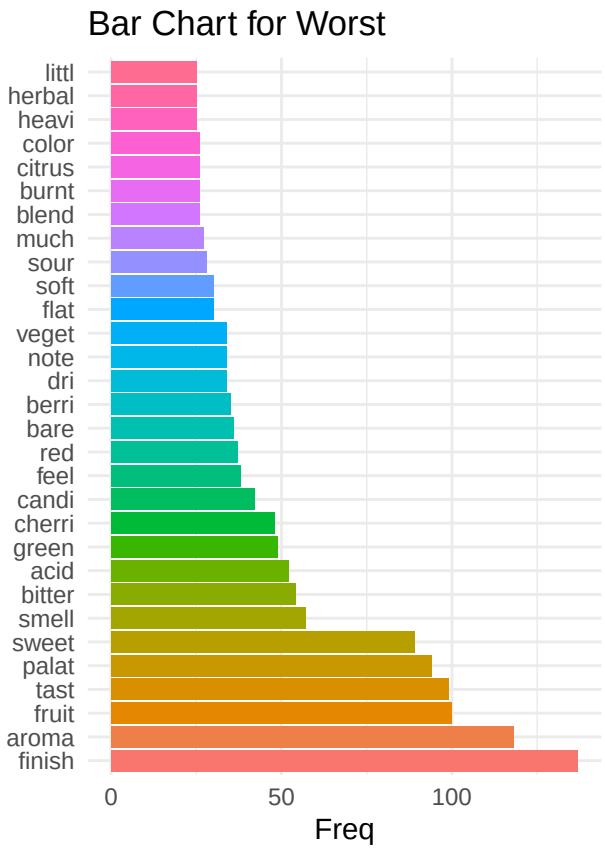
```
revG <- dfG$description
revB <- dfB$description
obviousWords <- c("wine","flavors","now","nose","well","show","like","aroma","drink","body","offer","vi
stopwords = c(stopwords::stopwords("en"),tolower(df1$variety),df1$province,df1$country,obviousWords)
#wds <- unlist(tokenize_ngrams(rev,n=2,n_min=1,stopwords=stopwords))
wG<-unlist(tokenize_word_stems(revG,stopwords=stopwords))
wB <- unlist(tokenize_word_stems(revB,stopwords=stopwords))
wdsG <- data.frame(sort(table(wG),decreasing = T))
wdsB <- data.frame(sort(table(wB),decreasing = T))

#Storing plots in variable in order to use grid.arrange
#Bar plot will be used to more accurately view the frequency of each word
#Only the top 30 words are used for ease of visualization
Gp <- ggplot(wdsG[1:30,],
       aes(x=wG, y=Freq,fill=wG))+
  geom_bar(stat="identity") +
  guides(fill = F)+
  xlab(NULL)+
  coord_flip()+
  labs(title = "Bar Chart for Best")+
  theme_minimal()
```

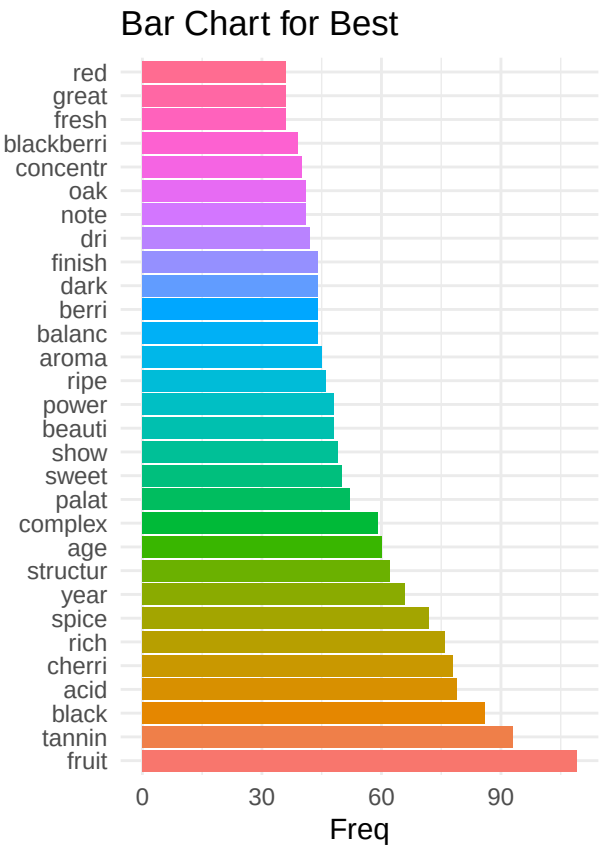
```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
Bp <- ggplot(wdsB[1:30,],
       aes(x=wB, y=Freq,fill=wB))+
  geom_bar(stat="identity") +
  xlab(NULL)+
  guides(fill = F)+
  coord_flip()+
  labs(title = "Bar Chart for Worst")+
  theme_minimal()
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
grid.arrange(Bp,Gp,ncol=2)
```





Firstly it is important to note that there is more words in the "best" category, however we are interested in the most frequently used words in each group, and aren't comparing the actual word frequencies between groups.
It was very interesting comparing the words used in the best wines in comparison to the worst wines (relative in the datset).

When describing the worst wines, they seem to focus more on more general terms, fruit, taste, smell, bitter, cherry, aroma. However the words that are used to describe the best wines are more precise, poetic and sophisticated. Such as, rich, structure, beautiful, power, balanced, complex.

Tannin which is top 2nd most frequent used word in the best wines is not used in the worst wines. Tannin is a very important concept when it comes to wine in general. Maybe tannins isn't as important for lesser quality wines but is essential for quality wines.

Moreover, the words age and year were used quite frequently when describing the best wines. Whereas it wasn't used much at all in the worst wines. Meaning that the age of the wine (most likely how old it is) is an important factor when it comes to the quality of the best wines. This is also might link up with tannins because wine aging has a significant effect on the tannins of the wine.

```
#clear memory
rm(df1)
rm(df)
rm(df10)
rm(df102)
rm(wdsdf)
```

Part 2

Tokenizer package analysis

As a follow on from Part 1 the package that will be analyzed is tokenizers. Tokenizers' functions, as explained earlier, takes in a text of data and returns a vector of words from the text.

```
library(tidytext)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::combine() masks gridExtra::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
library(ggplot2)
library(stopwords)
library(tokenizers)
citation("tokenizers")
```

```
##
## To cite the tokenizers package in publications, please cite the paper
## in the Journal of Open Source Software:
##
##   Lincoln A. Mullen et al., "Fast, Consistent Tokenization of Natural
##   Language Text," Journal of Open Source Software 3, no. 23 (2018):
##   655, https://doi.org/10.21105/joss.00655.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {Fast, Consistent Tokenization of Natural Language Text},
##     author = {Lincoln A. Mullen and Kenneth Benoit and Os Keyes and Dmitry Selivanov and Jeffrey Arn
##     journal = {Journal of Open Source Software},
##     year = {2018},
##     volume = {3},
##     issue = {23},
##     pages = {655},
##     url = {https://doi.org/10.21105/joss.00655},
##     doi = {10.21105/joss.00655},
##   }
```

The dataset that will be used is from kaggle, and contains movie titles, their genre and their plots. We will be using the tokenizers' functions to examine in more detail the different movie genres.

Link to dataset - https://www.kaggle.com/jrobischon/wikipedia-movie-plots/
The first part to this analysis is to compare its speed to a similar function from the tidyverse packages called unnest_tokens.

```
#Read in the data
df0 <- read.csv("wiki_movie_plots_deduped.csv")
df0 <- df0[,c('Title','Genre','Plot')]

#Time difference analysis from unest_tokens a tidyverse function
startTy <- Sys.time()
word_df1<-unnest_tokens(df0,word,Plot)%>% # unnest automatically has stop words
  anti_join(stop_words)%>%
  count(word,sort=TRUE) # count words function
```

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

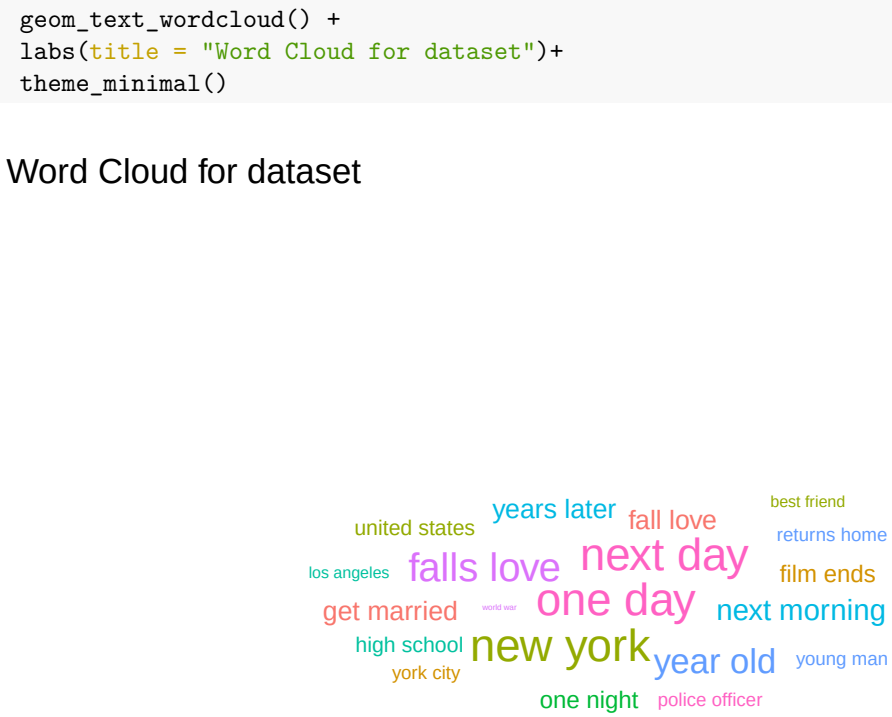
```
endTy <- Sys.time()

#In order to get the same output as above, we must unlist the tokens
#this is because toniz_words inputs is a list, indeces are the dataframe indeces
#unnest_tokens output is a vector
#table() used to count up
startTo <- Sys.time()
#need to specify the stop words
word_df2<-unlist(tokenize_words(df0$Plot, stopwords = stopwords::stopwords("en"),strip_numeric = TRUE))
word_df2<-data.frame(sort(table(word_df2),decreasing=TRUE))
endTo <- Sys.time()




print(paste("It took the tidytext package", round(endTy - startTy,2),"seconds","to perform tokenisation
```

```
## [1] "It took the tidytext package 11.29 seconds to perform tokenisation and a count of the tokens wh
```

This shows the speed of the tokenizer package. On top of this, it has other useful functions than the word tokenization, some which tidytext does not have.
For example:/ You can tokenize group of words,
tokenize_ngrams(words, n = 5, n_min = 2), grouping with at most n words and at least n_min words.

```
#isolate romace themded genre, need to use grepl as many types that isnt just romace, ie romance and co
dfc <- df0[grepl("romance",df0$Genre,fixed=TRUE),]

word_df<-unlist(tokenize_ngrams(df0$Plot, n=2, n_min=2,stopwords = stopwords::stopwords("en")))#,strip_
word_df<-data.frame(sort(table(word_df),decreasing=TRUE))

#plotting a word cloud of the top 20 words
ggplot(word_df[1:20,],
       aes(label=word_df,color = factor(sample.int(10, 20, replace = TRUE)),size=Freq))+
```

```
geom_text_wordcloud() +
labs(title = "Word Cloud for dataset")+
theme_minimal()
```



The analysis performed above using the ngram function is useful. for example, if you analysed 1 word tokens, you may find a high number of "new" tokens and "york" possibly as well. These tokens when taken independently are meaningless, it is only when you put them together that you can form some sort of conclusion, ie. a lot of romance movies happen in New York.

Tokenizing Tweets

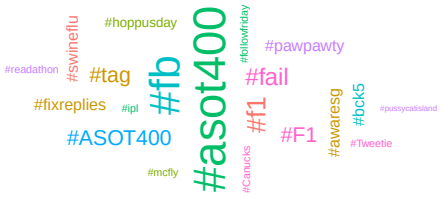
Interestingly the package also provides a useful function specialized in analysing tweets. Usually the tokenize_words function preserve only the word and ignores and symbol attached to it, such as full stops, commas or in the case of tweets, hashtags. Therefore this function preserves the hashtag and the "@" symbol too. This means #rstudio and rstudio are treated as different tokens.

Tweet Analysis

In order to showcase this function a new dataset is loaded containing 1600000 tweets (https://www.kaggle.com/kazanova/sentiment140). The exact information about this dataset was unable to be found. However only the dates and the tweets are of importance, as it is only being used to showcase the tokenise_tweets function and show why it is useful.

```
library(stringi)#need this package in order to extract the day and the years
#reading in the data
df0 <- read.csv("~/training.1600000.processed.noemoticon.csv")
#For computer power reasons only looking at 100 thousands tweets
l1 <- df0[1:100000,c(3,6)]#Want to keep the dates and the tweets
#renaming columns and adding a year and a day column
names(l1)<-c("fulldate","tweet")
l1$day<-str_extract(l1$fulldate, regex("[\\w]{3}\\s[\\w]{3}\\s[\\d]{2}"))#interested to look at the day
l1$year<-str_extract(l1$fulldate,regex("[\\d]{4}"))




#tokenising function
words<-tokenize_tweets(l1$tweet)
#count tokens
word_df<-data.frame(sort(table(unlist(words)),decreasing=TRUE))
names(word_df)<-c("word","Freq")

#This time we are only tooking at the frequency of hashtags used.
dfH <- word_df[grepl("#",word_df$word,fixed=TRUE),]
#Plotting word cloud with some words and 90deg angle
angle = 90 * sample(c(0, 1), dim(dfH)[1], replace = TRUE, prob = c(60, 40))
dfH<-cbind(dfH,angle)
#plotting
ggplot(dfH[1:20,],
       aes(label=word
           ,color = factor(sample.int(10, 20, replace = TRUE)),size=Freq, angle = angle))+
  geom_text_wordcloud() +
  labs(title = "Word Cloud for dataset")+
  theme_minimal()
```

Word Cloud for dataset



These are the 20 most used hashtags, in the 100 thousands tweets of the dataset. #asot400 (and #ASOT400, tokenization for hashtags is case sensitive) seemed to have been a popular hashtag, similar with #f1.

In this next part we will go into more detail and complete this analysis, we will uncover what is #asot400 and in doing so will uncover the true usage and potential application of tokenize tweets.

```
#start by doing a grep search and to find what dates this hash tag was used

asot <- l1[c(grepl("#ASOT400",l1$tweet,fixed=TRUE),grepl("#asot400",l1$tweet,fixed=TRUE)),c("day","year
#store the days that #asot was tweeted
days <- unique(asot[complete.cases(asot),"day"])
#how many tweets made in those days
tdays <-length(l1$day[which(l1$day%in%days)])

year <- unique(asot[complete.cases(asot),"year"])
print(paste("there were",dim(asot)[1],"from",tdays,
            "(about",round((dim(asot)[1]/tdays)*100,2),"percent)","tweets made on the days:"))
```

```
## [1] "there were 137 from 21711 (about 0.63 percent) tweets made on the days:"
```

```
print(paste(days,year))
```

```
## [1] "Fri Apr 17 2009" "Sat Apr 18 2009" "Sun Apr 19 2009"
```

The #asot400 was used on 3 consecutive days in 2009. After a quick Google search, it was found that

ASOT stands for "a state of trance" a popular radio show which carried out a 72 hour broadcast from 17th to the 19th as a celebration for its 400th transmissions (https://www.astateoftrance.com/news/asot-400/).

In conclusion the tokenize_tweets function shows that the tweet tokenisation and in this case analysing the hashtags works very similarly to the "Trending" function on Twitter. "Trending" on Twitter analyses all tweets written and groups them into tweets that are "about" the same thing. If there are a large volume of tweets about the same subject in a small time frame then they appear on the "Trending" tab. In this case for simplicity and to explain the tokenenize_tweet function, it was easy to uncover the most used hashtags and find out what is "trending" based on the hashtags used.

```
#Clearing memory
rm(df0)
rm(dfc)
rm(words)
rm(word_df1)
rm(word_df2)
```

Part 3

The third part builds on from the tokenization packages.
Tokenization is usually the first step to sentiment analysis. Below is the code that attempts to take a tokenized object and output whether the sentiment is negative, positive or neutral for that tokenized object. The function is simple and has two layers. Firstly it counts up all the positive and negative words that are found in the lexicon that was constructed by Minqing Hu and Bing Liu.
A lexicon is a group of words, such as positive words, that have been grouped together in a vector.

The next level is that if there is a "not" before the positive or negative word, it count it as the opposite.
It is possibly to do this without a for loop because, the input to the sentiment function is the output to the tokenize_ngrams(n=2,n_min=1), which as explained earlier, tokenizes for every word, the word itself and the word plus the word before it, ie. groups at lest one word and at most 2 words.
Therefore if "not" + positive word was found in the tokenzed object then it is counted as a negative word.

Finally if there are more postive words the function returns 1, which scores a positive sentiment.
If there are more negative words the function returns -1, which scores a negative sentiment.
If there are equal or none at all it retruns 0, which scores a neutral sentiment.

```
#reading in the lexicons and the citation
posw <- scan("~/Masters/RProg/lexicon/positive-words.txt",what="character",sep="\n")
negw <- scan("~/Masters/RProg/lexicon/negative-words.txt",what="character",sep="\n")
cit <- scan("~/Masters/RProg/lexicon/citation.txt",what="character",sep="\n")
print(cit)
```

```
##  [1] "This file and the papers can all be downloaded from"
##  [2] "   http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html"
##  [3] "  "
##  [4] "  If you use this list, please cite the following paper:"
##  [5] "  "
##  [6] "    Minqing Hu and Bing Liu. \"Mining and Summarizing Customer Reviews.\""
```

```
##  [7] "        Proceedings of the ACM SIGKDD International Conference on Knowledge"
##  [8] "        Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle,"
##  [9] "        Washington, USA,"
## [10] "  "
## [11] "  Notes:"
## [12] "     1. The appearance of an opinion word in a sentence does not necessarily"
## [13] "        mean that the sentence expresses a positive or negative opinion."
## [14] "        See the paper below:"
## [15] "  "
## [16] "        Bing Liu. \"Sentiment Analysis and Subjectivity.\" An chapter in"
## [17] "           Handbook of Natural Language Processing, Second Edition,"
## [18] "           (editors: N. Indurkhya and F. J. Damerau), 2010."
## [19] "  "
## [20] "     2. You will notice many misspelled words in the list. They are not"
## [21] "        mistakes. They are included as these misspelled words appear"
## [22] "        frequently in social media content."
```

Sentiment scoring function

```
#not vector
Not <- c("not")

# function sentiment scoring
score <- function(v){ #Imoprtant to make sure input is a vector not a list
  #words <- list()

  #save pos and neg words in variable
  pw <- v[which(v%in%posw)]
  nw <- v[which(v%in%negw)]


  #look for the not + postive and not + neg words
  NplusP <- v[which(v%in%paste(Not,pw,sep=" "))]
  PplusN <- v[which(v%in%paste(Not,nw,sep=" "))]



  if (length(NplusP)>0){##are there any not +
    dblNP <-unlist(strsplit(NplusP,split = " "))#split the list, sequence is always the same seebelow
    dblNP <- dblNP[seq(2,length(dblNP),2)]#seq is "not, something, not, something, etc"

    #This means for every not + word there is the NplusP variable there's is the same word in pw variab
    #That word in pw is a false positive
    #This part gets rid of that false positive word
    match <- table(pw[which(pw%in%dblNP )])
    tab <- table(dblNP)
    trueP <- match - tab

    pw <- c(pw[which(pw%in%dblNP==FALSE)],
            rep(names(trueP)[which(trueP>0)],trueP[which(trueP>0)]))

    nw <- c(nw, v[which(v%in%NplusP)])
```

```
  }
  #Same process repeated for false negatives
  if (length(PplusN)>0){
    dbl <-unlist(strsplit(PplusN,split = " "))
    dbl <- dbl[seq(2,length(dbl),2)]#seq is "not, something, not, something etc"


    match2 <- table(nw[which(nw%in%dbl )])
    tab2 <- table(dbl)
    trueN <- match2 - tab2

    nw <- c(nw[which(nw%in%dbl==FALSE)],
            rep(names(trueN)[which(trueN>0)],trueN[which(trueN>0)]))


    pw <- c(pw, v[which(v%in%PplusN)])
  }

  #How many  pos and neg words
  num_pw <- length(pw)
  num_nw <- length(nw)

  #Scoring, subtraction to see which theres most
  if (num_pw-num_nw >0){
    score <- 1
  }else if (num_pw-num_nw < 0){
    score <- -1
  }else{
    score <- 0
  }


  result <- c("score"=score,"diff" = num_pw-num_nw)


  class(result) <- c("sentiment","explain")
  all <- list(result,pw,nw)
  class(all)<-"explain"

  # classes for summary, prints and plots

  return(all)
  #return the result which contains, scoring(-1,-0,1)
  #and diff (number of pos words - number of neg words)
  #Also returns the lits of positive words and negative words for that tokenized object
}


score.sentiment <- function(lst){#simply returns only the score -1,0,1
  return(score(lst)[[1]]["score"])
}

print.explain <- function(lst){#prints a bit more detailed summary of the output
```

```
  if(lst[[1]]["score"] == 1){
    word <- "positive"
  }else if(lst[[1]]["score"] == -1){
    word <- "negative"
  }else{
    word <- "neutral"
  }
  cat("The sentiment is ", word,"\n")

  if(length(lst[[2]])!=0){
    cat("The list of positive words are :","\n",
        lst[[2]],"\n")

  }else{
    cat("There are no positive words","\n")
  }
  if(length(lst[[3]])!=0){
    cat("The list of negative words are :","\n",
        lst[[3]])
  }else{
       cat("There are no negative words","\n")
  }
}
```

Summary function

A summary function is created below.
This takes a data.frame as input. However in the data.frame it must contain a "year" variable and a score(-1,-0,1) variable (ie. the output to the sentiment scoring function)

```
#NGtok <- tokenize_ngrams(df$sentence, n= 2, n_min=1)
#$score <- unlist(sapply(NGtok,score.sentiment))
#class(df) <- "news"

summary.news <- function(df){
  if(length(unique(df$year))==1){
    year <- unique(df$year)
  }else{
    year <- paste(unique(df$year)[1],"to",unique(df$year)[length(unique(df$year))])
  }
  scores <- df$score
  avg <- mean(scores)
  std <- sd(scores)
  Npos <- length(scores[which(scores>0)])
  Nneg <- length(scores[which(scores<0)])
  Nneut <- length(df$score[which(scores==0)])

  cat("In ",year,"there was :","\n",
      Npos, "positive stories","\n",
      Nneg, "negative stories", "\n",
      Nneut, "neutral stories", "\n",
      "Mean sentiment score of:", avg, "\n",
      "Standard deviation of ", std,"\n",
```

```
      "----------------------","\n")

 perc <- ((Npos-Nneg)/(Npos+Nneg))*100

 if (Npos > Nneg){
   cat("There was ",round(perc,2),"% ","more positive stories")
   }else {
   cat("There was ",round(perc*-1,2),"% ","more negative stories")
   }

}

#summary(df)
```

Plotting function

A plotting function that better helps visuals the amount of positive and negative news/articles/text in a data.frame.
Same input as the summary function above.

```
plot.news <- function(df){
  if(length(unique(df$year))==1){
    year <- unique(df$year)
  }else{
    year <- paste(unique(df$year)[1],"to",unique(df$year)[length(unique(df$year))])
  }
  tab <- table(df$score)
  st <- sum(tab)
  for (i in 1:3){
    tab[i] <- round((tab[i]/st)*100,2)
  }
  names(tab) <- c("negative","neutral","positive")
  labels <- c("positive"=1,"negative"=-1,"neutral"=0)
  barplot(tab,
      col = c("red","yellow","green"),
      main = paste(year, "Bar Chart distribution of news sentiment"),#,
      ylab = "Percentage")
  lables <- c()

}
```

What Data will be Analysed?

The data was taken from kaggle, link is https://www.kaggle.com/tumanovalexander/nyt-articles-data./ This data set contains articles of various lengths from the New York Times (NYT). It ranges from 1920 to 2020, (only 1960 to 2020 will be used for this Analysis).
It has two cloumns, year and artcile.

Aim of the Anlysis

The aim of the analysis, is to use the sentiment scoring function on the Articles published in the NYT. This will mean for each article for every year a score of -1,0,1 will be given. Next we will average out the positive and negative scores for each article in year and give an overal sentiment score for that year.
The next part is to analyse whether the sentiment function works or not.
One could read the articles and assign a sentiment score by hand and compare results to the function's output. This would be the most effective way but it would be very time consuming as well so another approach will be conducted, whereby the sentiment score will be compared with another metric call the consumer confidence index (CII).

Reading in Data

The data was put in separate csv files according to each year. The easiest way to concatenate them would be to do it outside R. Using a cat command found on most bash terminals. It saves having to read them all in to R and rbind them.

```
cat ~/Masters/RProg/News_data/data/*.csv > ~/Masters/RProg/News_data/data/data.csv
```

```
df <- read.csv("~/Masters/RProg/News_data/data/data.csv")
df <- df[complete.cases(df),]
dim(df)
```

```
## [1] 8926419       3
```

```
names(df)
```

```
## [1] "X"        "year"     "sentence"
```

This dataset contains 8926419 rows and 3 columns. The column "sentence" contains a few sentences on an article published in the NWT. "X" columns are just indices and "year" column contains the year that article was published.

```
#how many articles in 2010
print(length(df$year[which(df$year=="2010",)]))
```

```
## [1] 169520
```

For example this dataset contains rlength(df$year[which(df$year==“2010”,)]‘ articles from 2010.

```
print(paste("The mean articles published per year is:",round(mean(table(df$year)),2),"with a standard d
```

```
## [1] "The mean articles published per year is: 146334.74 with a standard deviation of 43200.11"
```

We can see straight away that this data set has a big enough variance and this might lead to seom inaccuarcies.

```
print(paste("the years being studied for this analysis are",unique(df$year)[1],"through to 2020"))
```

```
## [1] "the years being studied for this analysis are 1960 through to 2020"
```

```
#frequency of Articles per year
tab <- table(df$year)
mn <- min(table(df$year))

print(paste("in the year",labels(tab[which(tab==mn)]), "there was", mn, "recorded articles in the datas
```

```
## [1] "in the year 1979 there was 20582 recorded articles in the dataset"
```

1979 is very under represented, in comparison to do the rest of the years (explained by the big standard deviation number of the data set). It may also lead to inaccuracies.

Sentiment scoring for each year

Because the dataset is so big, the sentiment function will be applied using a parallel process. This allows more than one CPU core to be used for the process and thus will be much quicker.

```
#parrellel function
library(doParallel)
```

```
## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loading required package: iterators

## Loading required package: parallel
```

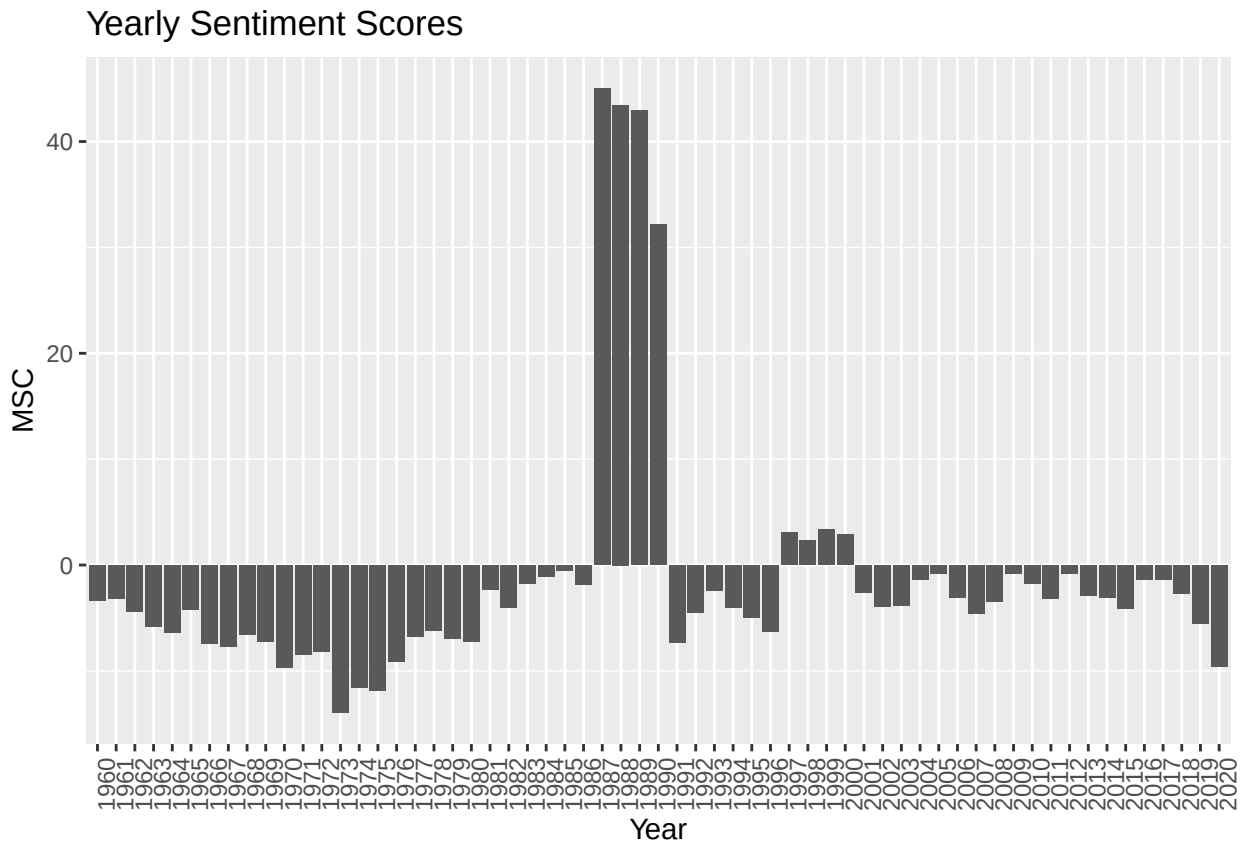
```
library(foreach)
```

```
cores <- detectCores()

cl <- makeCluster(6)#ores[1]/2)#cores[1]/2) #not to overload your computer, cores = 8
registerDoParallel(cl)



#combined the functions tokenezisation and score sentiment into the one
parr2 <- function(txt){
```

```
  NGtok <- unlist(tokenize_ngrams(txt, n= 2, n_min=1))
  return(score.sentiment(NGtok))
}


clusterExport(cl, varlist = c('score','posw','negw','Not','score.sentiment','parr2','tokenize_ngrams',"

s <- Sys.time()
df$score <- unlist(parSapply(cl,X = df$sentence,parr2))
df<- df[complete.cases(data.frame(df)),] # a variable "year" gets added to the year column, it has NAs
e <- Sys.time()

t <- round(e-s,2)
print(paste("It took",t,"min","to run the tokenization function and the sentiment scoring function on a
```

```
## [1] "It took 26.07 min to run the tokenization function and the sentiment scoring function on a data
```

```
#Giving it a class for my plot function
#class(df) <- "news"
#Scoring the none neutral
#df <- df[which(df$score!=0),]
#df1 <- df[which(df$score==0),]

#percentage differnce of positive to negative news, ie, -10% = 10% more bad news
pdiff <- function(x){
  return(round(mean(x)*100,2))
}
#By excluding the neutral news and getting the mean of positive to negative mean, effectively you're ju
dfs <- aggregate(x= df[which(df$score!=0),"score"], list(Year = df[which(df$score!=0),"year"]), pdiff)
dfs <- aggregate(x= df[,"score"], list(Year = df[,"year"]), pdiff)
names(dfs) <- c("Year","MSC")

ggplot(dfs, aes(x = Year, y = MSC))+
       geom_bar(stat="identity")+
  theme(axis.text.x = element_text(angle = 90))+
  ggtitle("Yearly Sentiment Scores")
```

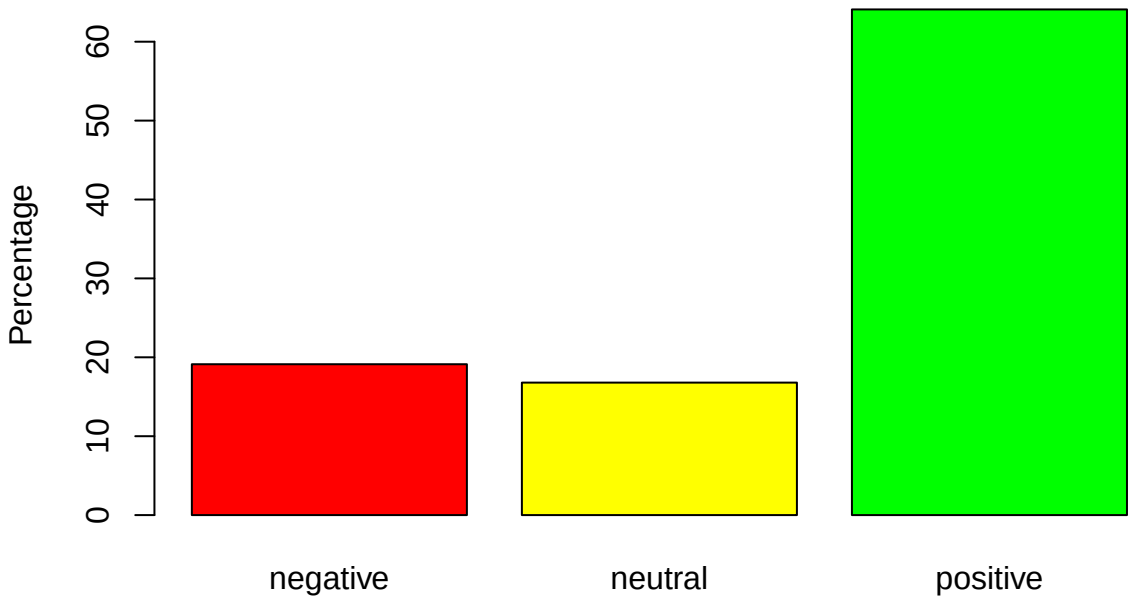


It seems that 1987 to 1990, are proportionally very positive in comparison to the other years. Reason is unknown and will be hard to deduce as there is so much data.
However, it seems to be accurate for 2020, as it has the most negative score in the last 2 decades. Which would make sense because of the pandemic.

```
#Show off plot and summary  function
df1987 <- df[which(df$year=="1987"),]
class(df1987) <- "news"
summary(df1987)
```

```
## In  1987 there was :
##  104763 positive stories
##  31251 negative stories
##  27439 neutral stories
##  Mean sentiment score of: 0.449744
##  Standard deviation of  0.7936393
##  ----------------------
## There was  54.05 %  more positive stories
```

```
plot(df1987)
```




```
rm(df1987)
```

```
#show off print function
set.seed(1)
ind <- runif(1)

df1987 <- df[which(df$year=="1987"&df$score == 1),]
sent <- df1987$sentence[ceiling(ind*length(df1987$sentence))]
sent <- tokenize_ngrams(sent,n =2,n_min=1)
print(df1987$sentence[ceiling(ind*length(df1987$sentence))])
```

```
## [1] "LEAD: After two years of negotiation, the state has approved a Boston plan to integrate all new
```

```
print("--------")
```

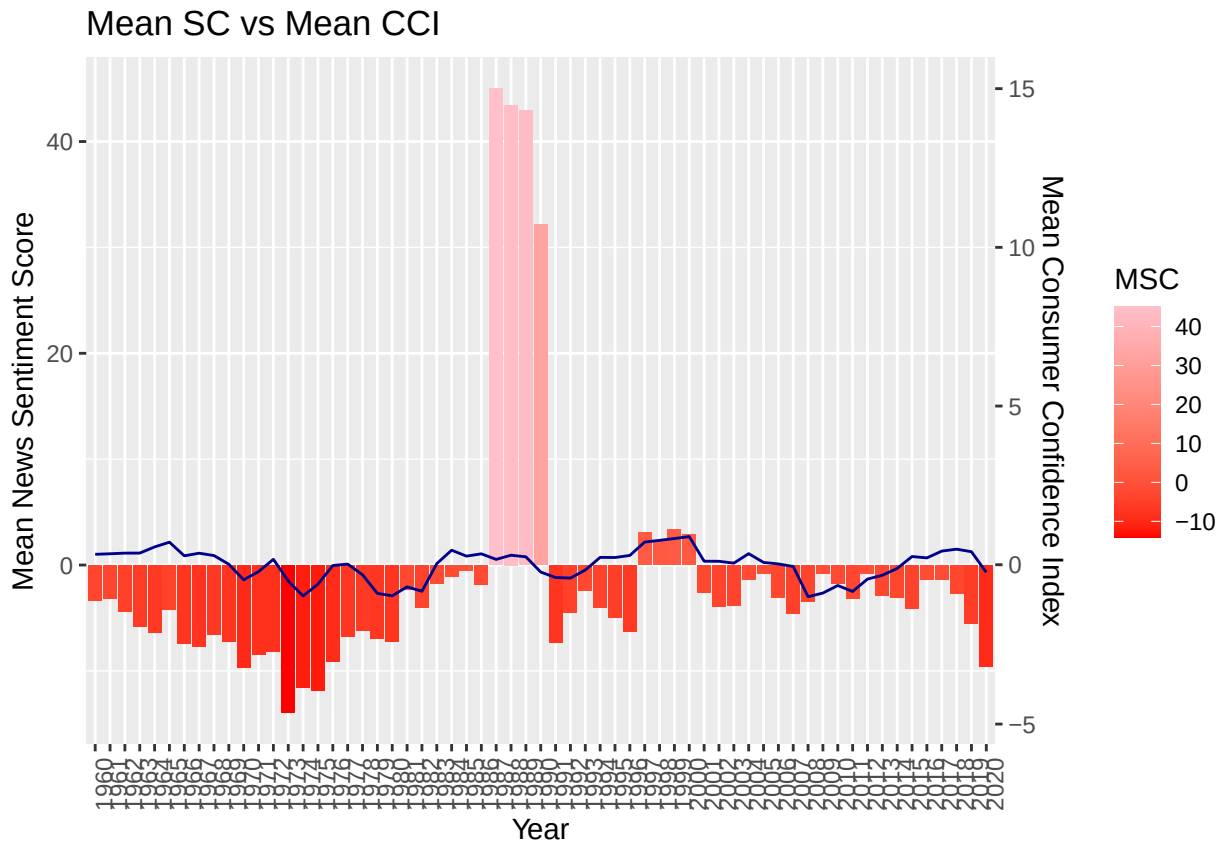
```
## [1] "--------"
```

```
print(score(sent[[1]]))#print.explain
```

```
## The sentiment is  positive
## The list of positive words are :
##  lead significant
## There are no negative words
```

To get a better understanding of the sentiment score result and to uncover whether it is truly accurate. The mean sentiment score (MSC) for each year will be plotted against the Consumer Confidence Index in America. This is an index that is based off a survey, asking question on economic stability but also their confidence on the economy as a whole. https://data.oecd.org/leadind/consumer-confidence-index-cci.htm/

```
#load in consumer confidence index

dfc <- read.csv("DP_LIVE_12122021173345665.csv")
dfc <- dfc[which(dfc$LOCATION=="USA"),c("TIME","Value")]
dfc$year <- gsub("-\\d+","",dfc$TIME)#create a column onfor the year



dfm <- aggregate(x= dfc$Value, list(Year = dfc$year), mean)
names(dfm)<-c("Year","CII")


dfplot <- merge(dfs,dfm,by="Year")#combine CII and MSC


p1<-ggplot(dfplot, aes(x=Year,group=1,fill=MSC))+ #group=1 needed otherwise geom_line wont work
  geom_bar(aes(y=MSC),stat="identity")+

  scale_fill_gradient(low="red", high="pink")+
  geom_line(aes(y=CII-100),color = "darkblue")+#-100 so that the graphs align and according the website
  scale_y_continuous(
    name = "Mean News Sentiment Score",
    sec.axis = sec_axis(~ ./3, name = "Mean Consumer Confidence Index"))+
    theme(axis.text.x = element_text(angle = 90)) +
    labs(color = "Legend")+
  ggtitle("Mean SC vs Mean CCI")


p1
```

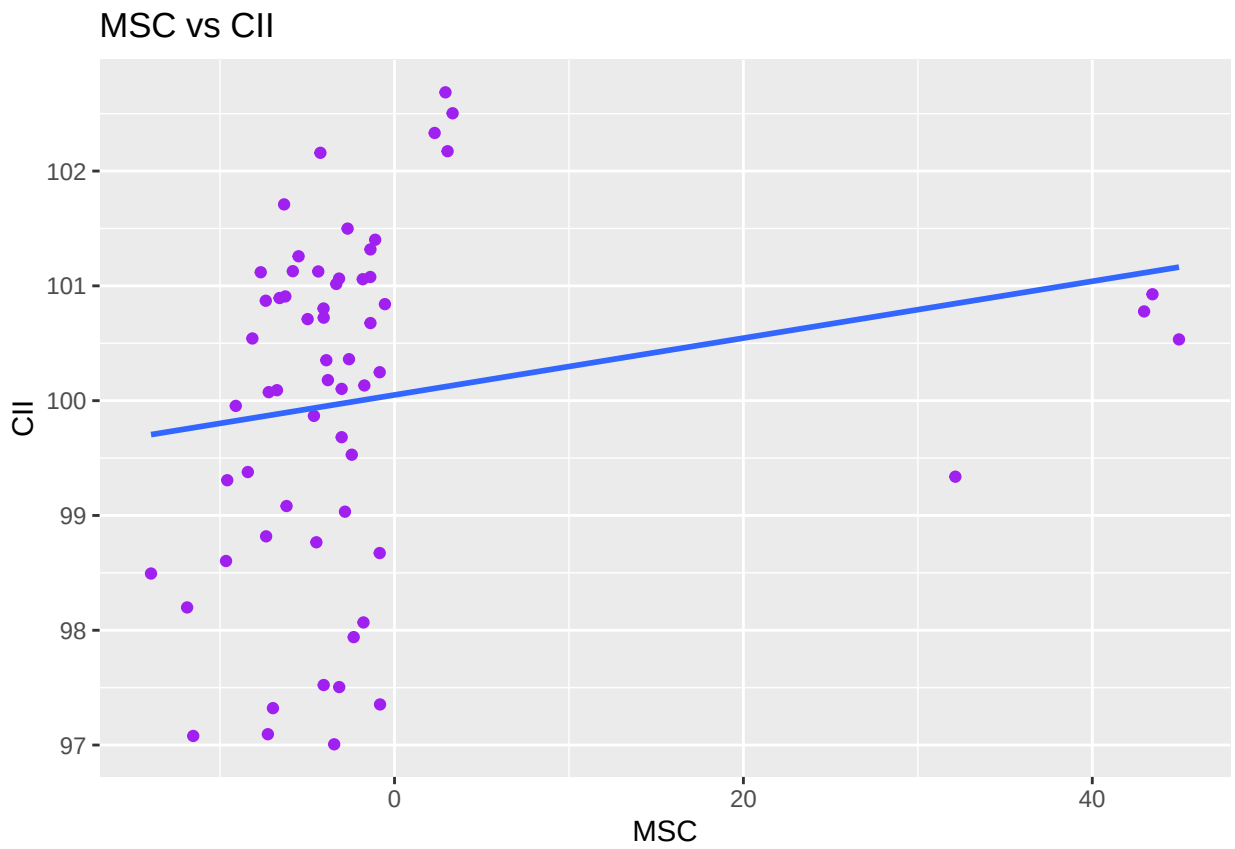


There seems to be some correlation between the two but it is hard to know if the correlation is random or not.
For example, the CII is positive in most cases when the Mean Sentiment Score (MSC) is positive as well or when the MSC is relatively quite low. This hold true for years 1987-1989, even though thise 3 years are disproportionately positive, the CII is also positive.
Moreover a sharp decrease in MSC is often followed by a decrease in CII. For example, in 2020, the CII isn't significantly lower compared to previous years, however the CII decreases sharply downward from decade all time high CII score. Lastly, the trend isn't so strong in the 1960s with high CII scores and low MSC.

In order to more accurately determine the relationship between the two, ie CII and MSC, 2 linear regressions will be conducted. One with all the data and the second the years 1987 to 1990, will be treated as outliers and omitted from the analysis.

```
years <- c("1987","1988","1989","1990")

ggplot(dfplot, aes(x=MSC,y=CII))+
  geom_point(color = "purple") +
  geom_smooth(method="lm", se=F)+
  labs(title = "MSC vs CII")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

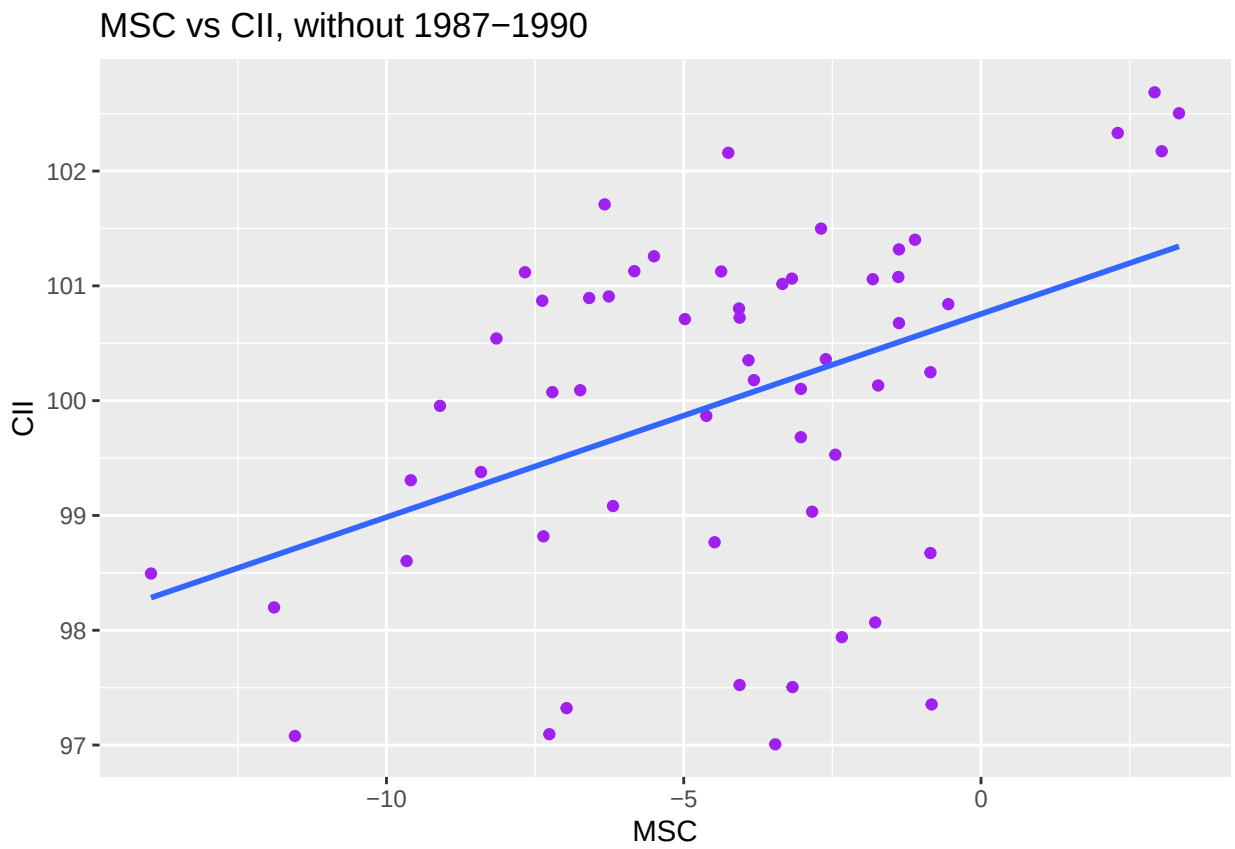



```
print(paste("coefficient of determination is", summary(lm(MSC~CII,data=dfplot))$r.squared))
```

```
## [1] "coefficient of determination is 0.0397571324573851"
```

```
ggplot(dfplot[which(dfplot$Year%in%years==FALSE),], aes(x=MSC,y=CII))+
  geom_point(color = "purple") +
  geom_smooth(method="lm", se=F)+
  labs(title = "MSC vs CII, without 1987-1990")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
print(paste("coefficient of determination is", summary(lm(MSC~CII,data=dfplot[which(dfplot$Year%in%years
```

```
## [1] "coefficient of determination is 0.178501252933335 when excluding 1987-1990"
```

When conducting a linear regression the association seems to be stronger if you treat 1987-1990 as outliers. However once this was done the association remained weak.

The question now lies whether CII is a good representation for sentiment analysis when looking at the general news as a whole?
For example CII looks at the economic confident and stability of consumers, now of course that depends on the news but more so economic , political news and foreign affairs, less so on sports news. This is common knowledge.
Whether the news data set contains accurate random selection sampling of news articles for every year?
This is only preliminary analysis on the function being used, there are no hard evidence to say that it doesn't work.
If there was a follow on analysis, a few data cleaning steps will have to be conducted first. Which would include filtering only economical, political news and foreign affairs. Then randomly choosing a pre determine number of articles from each year and making sure that they all have at least 2 sentences and at most 5. Then repeat the same procedure as before.

Thank you for giving me the chance to present to you my project.

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot. e