Code ▾

R Tutorial on Natural Language Processing

Julia Deaver, Julia Burek, Madeleine Jones, Chelsea Le Sage

2022-12-07

Background on Natural Language Processing

Natural Language Processing (NLP) refers to the branch of artificial intelligence that gives computers the ability to understand text and spoken words similar to how human beings can. NLP drives computer programs that translate text from one language to another, respond to spoken text, and summarize large volumes of text quickly. NLP also has major implications in solutions that help streamline business operations, increase employee productivity, and provide important insights. A particular branch of Natural Language Processing that we want to explore in this tutorial is Sentiment Analysis. Sentiment Analysis is an essential business for uncovering data insights from text. Sentiment analysis can analyze language used in social media posts, reviews, and more to extract attitudes and emotions in response to something such as a product. Businesses can then take these insights and make important decisions that help their company as a whole.

Another topic that is helpful with Natural Language Processing is TF-IDF (term frequency-inverse document frequency). It is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. It is a very useful technique in scoring words in machine learning algorithms for NLP. In order to build some kind of machine learning algorithm, it is helpful to transform words into numbers. This is why TF-IDF is useful so that an algorithm can understand the text data. The TF-IDF approach that we will explore in this tutorial is keyword extraction, which extracts the highest scoring words. By transforming textual data into numerical data, we are able to perform machine learning techniques such as k-means clustering. As we know k-means clustering analysis groups similar data points together to discover underlying patterns. K-means clustering can be very helpful when identifying underlying themes and patterns among different phrases.

Loading Packages

We will begin by importing the following packages which will be used throughout the tutorial. If you do not already have a package loaded, use the install.packages('*package name*') function first to install the package and then load it.

Hide

```
library(tidyverse)
library(stopwords)
library(tm)
library(dplyr)
library(stringi)
library(tidytext)
library(wordcloud)
library(ggplot2)
library(tidyr)
library(mgsub)
library(scales)
library(stringr)
library(widyr)
library(gridExtra)
library(factoextra)
```

Import the Data

For the course of this tutorial, we will be using a data set from Kaggle that records wine reviews. Download the dataset directly from Kaggle (linked in the README) or decompress the .zip file found in the repository.

Read in the .csv file as done below. There are 14 columns in this dataframe and 129971 rows. We drop the unnecessary indexing column, X, as we read in the data. This is performed by the `select(-X)` operation. We should observe 13 variables in our data now.

Hide

```
wine_reviews = read.csv("winemag-data-130k-v2.csv") %>% select(-X)
```

The names of each of the columns are below:

Hide

```
names(wine_reviews)
```

```
##  [1] "country"               "description"           "designation"
##  [4] "points"                "price"                 "province"
##  [7] "region_1"              "region_2"              "taster_name"
## [10] "taster_twitter_handle" "title"                 "variety"
## [13] "winery"
```

As mentioned in the README, `description` is our column of interest for NLP, as it provides tasting notes and flavor profiles for each observation. We isolate the `description` column and remove all punctuation, which would interfere with our language processing:

Hide

```
wine.reviews.text <- wine_reviews %>%
  select(description)
wine.reviews.text$description<-removePunctuation(wine.reviews.text$descriptio
n)
```

Now we're ready to start working with the data!

Exploratory Text Analysis (ETA)

By performing ETA we can gain further understanding of our data and explore interactions between our attributes. Notice that the `ignore.words` variable below specifies words to ignore while searching over our data. These are words that aren't meaningful or wouldn't provide any insight to our analysis, so we disregard them.

Hide

```
ignore.words <- data_frame(word = c("wine","flavors","drink"))
```

We start ETA by constructing a word frequency table.

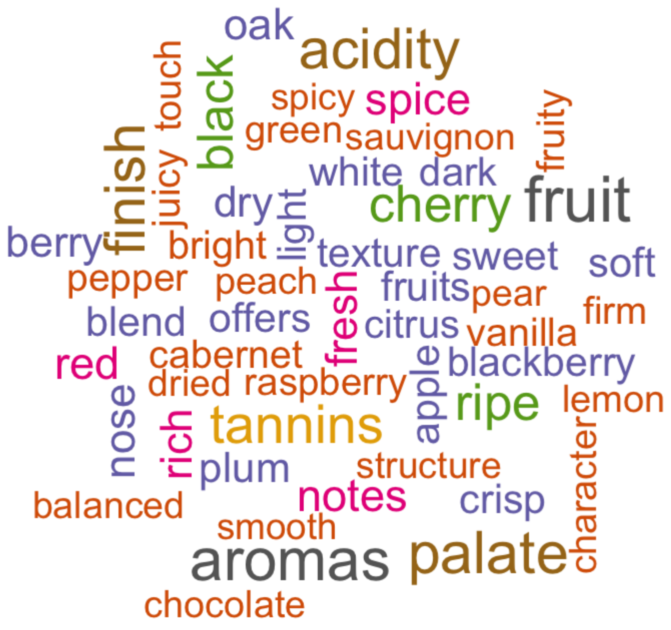
Hide

```
word.freq.table<- wine.reviews.text %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>%
  anti_join(ignore.words) %>%
  count(word, sort = TRUE)
```

While we could manually parse through the frequency table, a wordcloud produces a meaningful visual representation of word frequencies:

Hide

```
word.freq.table %>%
  filter(n>7000) %>%
  with(wordcloud(word, n,
                 scale = c(2,1),
                 colors = brewer.pal(8, "Dark2")))
```



It’s also interesting to explore how related these terms are to one another. We might hypothesize that wines described with “cherry” flavors also tend to be “sweet”. We build out the correlation between terms as follows:

Hide

```
wine.correlation.terms <- wine.reviews.text %>%
  mutate(review = row_number()) %>%
  unnest_tokens(word, description) %>%
  filter(!word %in% stop_words$word) %>%
  group_by(word) %>%
  filter(n() >= 5000)%>%
  pairwise_cor(word, review, sort = TRUE)
wine.correlation.terms
```

```
## # A tibble: 9,312 × 3
##    item1     item2     correlation
##    <chr>     <chr>           <dbl>
##  1 sauvignon cabernet        0.594
##  2 cabernet  sauvignon       0.594
##  3 merlot    cabernet        0.432
##  4 cabernet  merlot          0.432
##  5 sauvignon merlot          0.374
##  6 merlot    sauvignon       0.374
##  7 nose      palate          0.336
##  8 palate    nose            0.336
##  9 palate    aromas          0.319
## 10 aromas    palate          0.319
## # … with 9,302 more rows
```

To find a specific description pair you have in mind, simply index into the terms you're looking for! You can play around with passing in different strings to find specific correlation coefficients:

Hide

```
wine.correlation.terms[wine.correlation.terms$item1 == 'cherry' & wine.correl
ation.terms$item2 == 'sweet',]
```

```
## # A tibble: 1 × 3
##   item1  item2 correlation
##   <chr>  <chr>       <dbl>
## 1 cherry sweet    -0.00312
```

Lexicon Sentiment Analysis

Sentiment analysis is the process of identifying the sentiment of text by machine learning. There are three different sentiment lexicons contained in the tidytext package:

- AFINN
- NRC
- Bing

These sentiment lexicons are based on single English words in which each word is assigned a sentiment given the lexicon's particular scale. We will examine each of these sentiment lexicons and compare them using wines specifically from the United States, France, and Italy. These three countries comprise the majority of our data set with 54,504 wines coming from the United States; 22,093 from France; and 19,540 from Italy.

First, we will create 3 different data sets for each country's wines and keep only the description column. We will do this by filtering the rows of the original data set and selecting the `description` column.

Hide

```
us <- wine_reviews %>% filter(country == "US") %>% select(description)
france <- wine_reviews %>% filter(country == "France") %>% select(descriptio
n)
italy <- wine_reviews %>% filter(country == "Italy") %>% select(description)
```

Next, we will extract the individual words and word counts from each country's descriptions. We will do this for each country's data separately by tokenizing the words, removing the stop words, and then counting the occurrences of each word, similar to before.

Hide

```
us_words <- us %>%
  unnest_tokens(word, description)%>%
  anti_join(stop_words)%>%
  count(word, sort=TRUE)
```

Hide

```
france_words <- france %>%
  unnest_tokens(word, description)%>%
  anti_join(stop_words)%>%
  count(word, sort=TRUE)
```

Hide

```
italy_words <- italy %>%
  unnest_tokens(word, description)%>%
  anti_join(stop_words)%>%
  count(word, sort=TRUE)
```

AFINN

The first sentiment lexicon is AFINN, developed by Finn Arup Nielson. This lexicon analysis assigns sentiments to words based on a -5 to 5 scale, where a score of -5 indicates more extreme negative sentiment while a score of 5 indicates more extreme positive sentiment.

We will first calculate the sentiment using AFINN on each country's description words.

Hide

```
affin_us <- us_words %>%
  inner_join(get_sentiments("afinn")) #using a inner join to match words and
 add the sentiment variable

affin_france <- france_words %>%
  inner_join(get_sentiments("afinn")) #using a inner join to match words and
 add the sentiment variable

affin_italy <- italy_words %>%
  inner_join(get_sentiments("afinn")) #using a inner join to match words and
 add the sentiment variable
```

Next, we can plot a histogram of the AFINN sentiment results for each country for comparison:

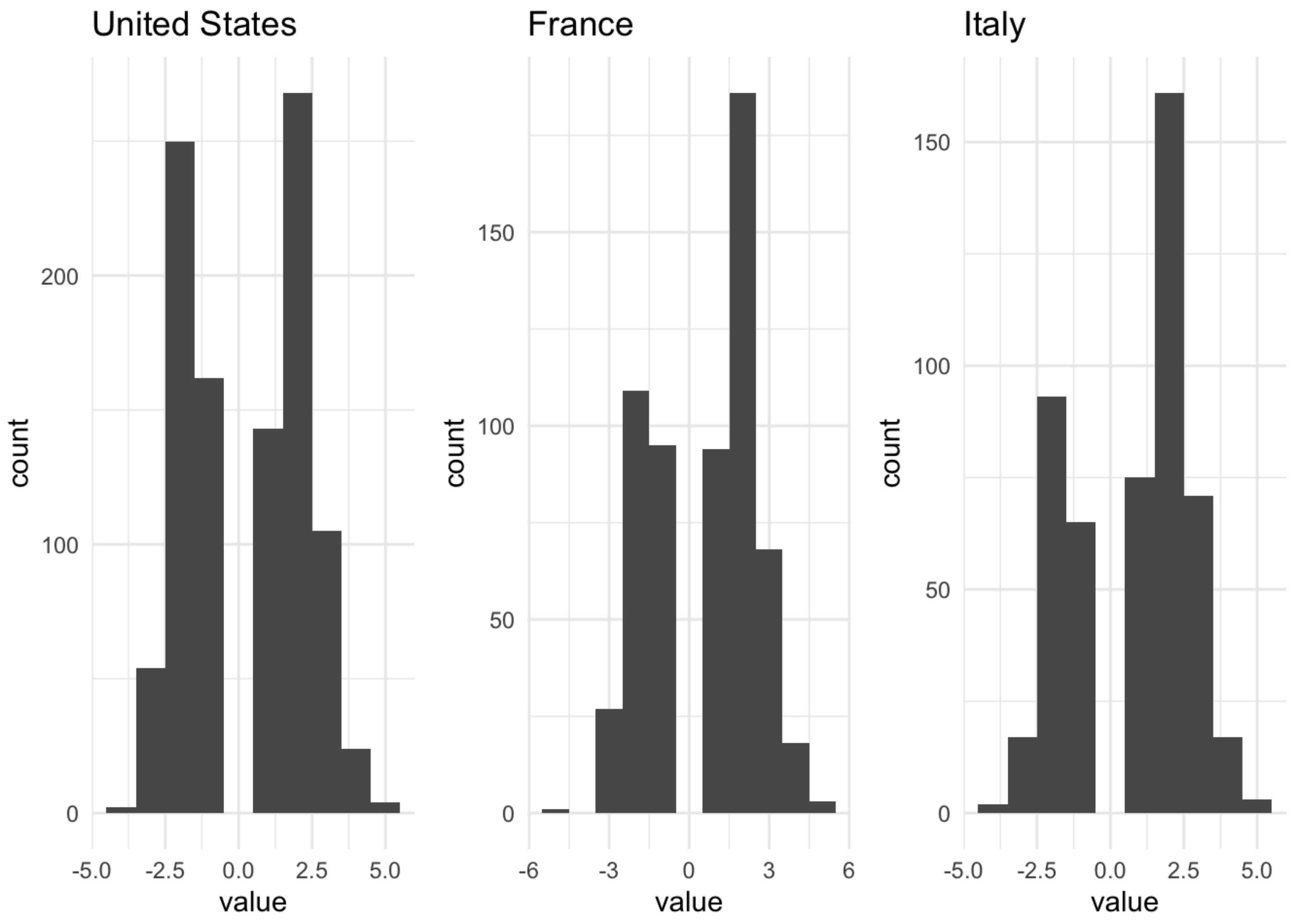
Hide

```
# create US plot
us_affin_plot <- ggplot(data = affin_us,
       aes(x=value))+
  geom_histogram(binwidth = 1)+
  ggtitle("United States")+
  theme_minimal()

# create France plot
france_affin_plot <- ggplot(data = affin_france,
       aes(x=value))+
  geom_histogram(binwidth = 1)+
  ggtitle("France")+
  theme_minimal()

# create Italy plot
italy_affin_plot <- ggplot(data = affin_italy,
       aes(x=value))+
  geom_histogram(binwidth = 1)+
  ggtitle("Italy")+
  theme_minimal()

grid.arrange(us_affin_plot, france_affin_plot, italy_affin_plot, ncol = 3)
```



Using the side-by-side comparison of the AFINN sentiment ranges for the three countries, we can see that the distribution of France and Italy wine descriptions appear relatively similar. Specifically, there appears to be a significantly larger spike at around 2 for both countries, indicating a positive word, compared to the peaks on the negative half of the scale which indicate negative words. In the US, however, the spikes on the positive and negative sides of the sentiment scale appear to be similar in magnitude. This suggests that the overall number of positive and negative wine descriptions may be closer in the US than the overall numbers in France or Italy.

Keep in mind that we have 54,504 US wines versus 22,093 from France and 19,540 from Italy, so the scale along the y-axis for the US graph is larger as these charts are built off of the count per dataset.

NRC

The second lexicon used is NRC, developed by Said Mohammad and Peter Turney. This sentiment analysis classifies words into categories such as positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. Though we would expect wine descriptions to be neutral, let’s see what emotions may be evoked!

We will now calculate the sentiment using NRC on each country’s description words.

Hide

```
# NRC on the US Data
nrc_us <- us_words %>%
  inner_join(get_sentiments("nrc"))

# NRC on the France Data
nrc_france <- france_words %>%
  inner_join(get_sentiments("nrc"))

# NRC on the Italy Data
nrc_italy <- italy_words %>%
  inner_join(get_sentiments("nrc"))
```

We can use the following code to extract the results of the NRC analysis into a data frame for comparison.

Hide

```
# extract the category labels
labels <- data.frame(table(nrc_us$sentiment))$Var1

# extract the counts of each category for each country
us_nrc_results <- data.frame(table(nrc_us$sentiment))$Freq
france_nrc_results <- data.frame(table(nrc_france$sentiment))$Freq
italy_nrc_results <- data.frame(table(nrc_italy$sentiment))$Freq

# put the results into a dataframe
nrc_results <- data.frame(cbind(us_nrc_results, france_nrc_results, italy_nrc
_results))
colnames(nrc_results) <- c("US", "France", "Italy")
rownames(nrc_results) <- labels

nrc_results
```

```
##                 US France Italy
## anger          333    162   130
## anticipation   431    292   239
## disgust        227    111    88
## fear           419    201   178
## joy            398    287   249
## negative       949    468   378
## positive      1237    854   754
## sadness        322    160   133
## surprise       265    170   143
## trust          573    400   340
```

The raw count of words in different sentiment categories may be hard to compare directly. Therefore, we can also convert these counts to percentages for each country in order to determine which country has the greatest percentage of words in each of the sentiment categories.

We can convert the raw counts to percentages using some calculations with the following code:

Hide

```
# convert each count to a percentage of the column total for each country
nrc_results$US <- round(nrc_results$US/sum(nrc_results$US) * 100, 2)
nrc_results$France <- round(nrc_results$France/sum(nrc_results$France) * 100,
2)
nrc_results$Italy <- round(nrc_results$Italy/sum(nrc_results$Italy) * 100, 2)

nrc_results
```

```
##                  US France Italy
## anger          6.46   5.22  4.94
## anticipation  8.36   9.40  9.08
## disgust        4.40   3.57  3.34
## fear           8.13   6.47  6.76
## joy            7.72   9.24  9.46
## negative      18.41  15.07 14.36
## positive      24.00  27.50 28.65
## sadness        6.25   5.15  5.05
## surprise       5.14   5.48  5.43
## trust         11.12  12.88 12.92
```

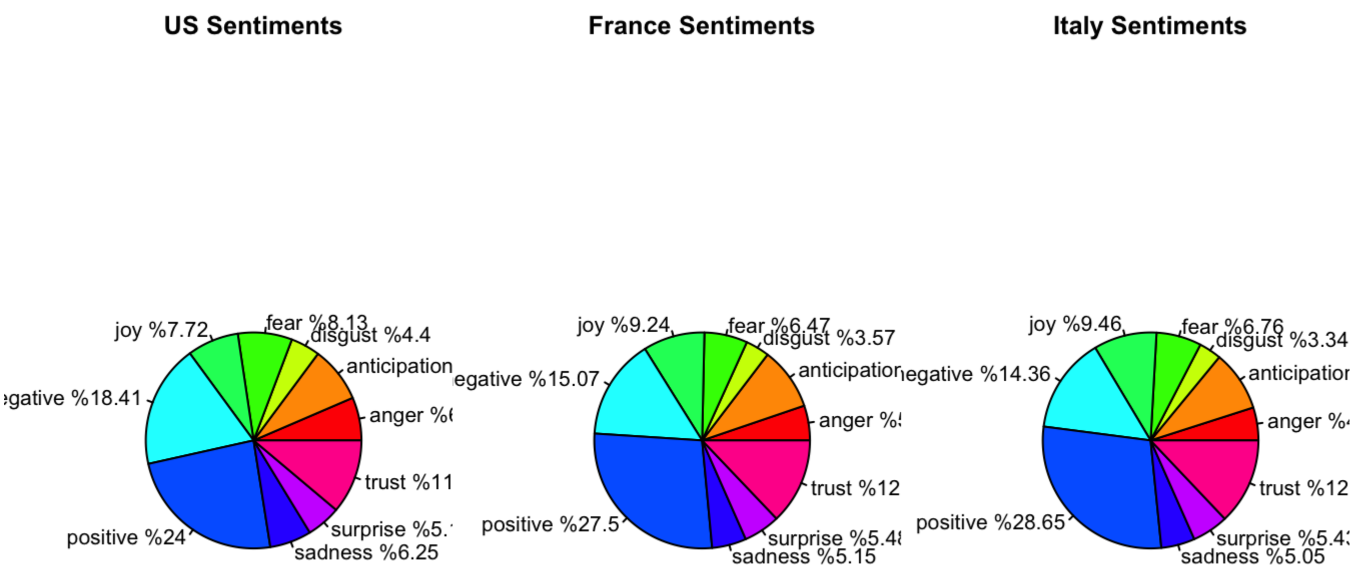
From this table, we can see that the rough percentages of each sentiment group is similar for the United States, France, and Italy. However, the Italy has the highest percentage of words with positive sentiment with 28.65%, followed by France with 27.5%, and then followed by the United States with 24%.

If we want to visualize these difference between countries, we can produce pie charts to see the distributions of sentiments per country of origin:

Hide

```
par(mfrow=c(1,3))

pie(nrc_results$US, labels = paste(labels, paste("%", nrc_results$US, sep=""
)), col=rainbow(10), main="US Sentiments")
pie(nrc_results$France, labels = paste(labels, paste("%", nrc_results$France,
sep="")), col=rainbow(10), main="France Sentiments")
pie(nrc_results$Italy, labels = paste(labels, paste("%", nrc_results$Italy, s
ep="")), col=rainbow(10), main="Italy Sentiments")
```



BING

Finally, the third lexicon is bing, developed by Bing Liu. Bing sentiment analysis classifies the sentiment of words as strictly positive or negative.

We can calculate the sentiment using Bing on each country's description words with the following code.

Hide

```
# calculate Bing on the US Data
bing_us <- us_words %>%
  inner_join(get_sentiments("bing"))

# calculate Bing on the France Data
bing_france <- france_words %>%
  inner_join(get_sentiments("bing"))

# calculate Bing on the Italy Data
bing_italy <- italy_words %>%
  inner_join(get_sentiments("bing"))
```

We can now use the following code to extract the results of the Bing analysis into a data frame for comparison.

Hide

```
# extract the counts of each category for each country
us_bing_results <- data.frame(table(bing_us$sentiment))$Freq
france_bing_results <- data.frame(table(bing_france$sentiment))$Freq
italy_bing_results <- data.frame(table(bing_italy$sentiment))$Freq

# put the results into a dataframe
bing_results <- data.frame(cbind(us_bing_results, france_bing_results, italy_
bing_results))
colnames(bing_results) <- c("US", "France", "Italy")
rownames(bing_results) <- c('negative', 'positive')

bing_results
```

```
##            US France Italy
## negative 1099    501   399
## positive 1056    716   630
```

Similar to with the NRC results, the raw counts of positive and negative words may be hard to compare directly. Therefore, we can also convert these counts to percentages for each country in order to determine which country has the greatest percentage of positive and negative words.

We can again convert the raw counts to percentages using the following code.

Hide

```
# convert each count to a percentage of the column total for each country
bing_results$US <- round(bing_results$US/sum(bing_results$US) * 100, 2)
bing_results$France <- round(bing_results$France/sum(bing_results$France) * 1
00, 2)
bing_results$Italy <- round(bing_results$Italy/sum(bing_results$Italy) * 100,
2)

bing_results
```

```
##          US France Italy
## negative 51  41.17 38.78
## positive 49  58.83 61.22
```

From the Bing percentage results, we again can determine the country with the most positive and negative reviews. Similar to the conclusions drawn from the previous analyses, Italy has the greatest percentage of positive reviews, followed by France, and then the United States.

Term Frequency - Inverse Document Frequency Analysis

For our last step of natural language processing, we will use Term Frequency - Inverse Document Frequency analysis, or tf-idf. Term frequency refers to how often a word appears in a single entity's text. Document Frequency refers to how often a word appears in a population of text.

We can use Term Frequency - Inverse Document Frequency analysis in order to analyze the most prevalent words, or phrases, unique to an individual country's wine descriptions as compared to the population of descriptions worldwide. For simplicity, we will again investigate the descriptions of the United States, France, and Italy.

We will first tokenize the words of each country's description into phrases of 2 words. This is because word pairings are likely to be more informative and provide more context than individual words on their own. We do this with the following code block:

Hide

```
tf_data <- wine_reviews %>%
  select(description, country) %>%
  unnest_tokens(phrase, description, token="ngrams", n=2)%>%
  group_by(country)%>%
  separate(phrase, c("word1", "word2"), sep=" ")%>%
  filter(!word1 %in% stop_words$word,
         !word2 %in% stop_words$word) %>%
  count(word1, word2, country, sort = TRUE)%>%
  ungroup() %>%
  unite(phrase, word1, word2, sep = " ")
```

We then apply the tf-idf algorithm on the phrases of 2 from every country and sort our results in descending order of term frequency over inverse document frequency.

Hide

```
tf_idf <- tf_data %>%
  bind_tf_idf(phrase, country, n)%>%
  arrange(desc(n))
```

Now, we can extract the top 5 most common phrases specific to the United States, France, and Italy.

Hide

```
tf_idf_us <- head(tf_idf %>% filter(country == "US"), 5)$phrase
tf_idf_france <- head(tf_idf %>% filter(country == "France"), 5)$phrase
tf_idf_italy <- head(tf_idf %>% filter(country == "Italy"), 5)$phrase


most_common_phrases <- data.frame(cbind(tf_idf_us, tf_idf_france, tf_idf_ital
y))
colnames(most_common_phrases) <- c('US', 'France', 'Italy')
rownames(most_common_phrases) <- c("Phrase 1","Phrase 2","Phrase 3","Phrase
 4","Phrase 5")
most_common_phrases
```

```
##                               US         France           Italy
## Phrase 1      fruit flavors black currant   palate offers
## Phrase 2       black cherry    red fruits    black cherry
## Phrase 3 cabernet sauvignon   fruity wine palate delivers
## Phrase 4         pinot noir fruit flavors    white pepper
## Phrase 5      medium bodied    wood aging   fresh acidity
```

There is not much overlap between the most common phrases for the countries; however, there are some common themes among them. France, for example, has 3 phrases that contain the word 'fruit' and Italy has two phrases that contain the word 'palate'. These common phrases are helpful in identifying the main differences across the reviews for the countries.

K-Means Clustering

Next, we will conduct K-Means clustering on the text descriptions of each wine in order to examine the differences between wines coming from the most prevalent wine-producing provinces.

First, we will use our td_idf algorithm for each country within the dataset. Will will begin by finding each unique country within the data set.

Hide

```
tf_idf1 <- tf_idf[tf_idf$country != "",]
countries<-unique(tf_idf1$country)
```

Hide

```
head(tf_idf1)
```

```
## # A tibble: 6 × 6
##   country phrase                 n      tf   idf  tf_idf
##   <chr>   <chr>              <int>   <dbl> <dbl>   <dbl>
## 1 US      fruit flavors       4170 0.00700 0.288 0.00202
## 2 Italy   palate offers       3547 0.0130  0.951 0.0124
## 3 Italy   black cherry        3245 0.0119  0.201 0.00239
## 4 US      black cherry        3194 0.00537 0.201 0.00108
## 5 US      cabernet sauvignon  3082 0.00518 0.417 0.00216
## 6 US      pinot noir          2278 0.00383 0.649 0.00248
```

We will now group each country and find the mean for the tf_idf values.

Hide

```
tf_idf2<-tf_idf1 %>%
  group_by(country) %>%
  summarise_at(vars(tf, idf, tf_idf),list(mean=mean))
```

Hide

```
tf_idf2 <- tf_idf2 %>% column_to_rownames(., var = 'country')
tf_idf2
```

```
##                            tf_mean idf_mean  tf_idf_mean
## Argentina              5.106991e-05 2.965020 1.181371e-04
## Armenia                7.692308e-02 1.974302 1.518694e-01
## Australia              7.089685e-05 2.943949 1.702110e-04
## Austria                5.556790e-05 3.076185 1.406401e-04
## Bosnia and Herzegovina 4.000000e-02 1.802920 7.211681e-02
## Brazil                 2.100840e-03 2.316951 4.579668e-03
## Bulgaria               1.049318e-03 2.209334 1.866497e-03
## Canada                 4.777831e-04 2.609479 1.135875e-03
## Chile                  4.507144e-05 2.979051 1.040230e-04
## China                  5.555556e-02 1.691157 9.395314e-02
## Croatia                1.351351e-03 2.334815 2.900878e-03
## Cyprus                 9.708738e-03 2.527374 2.406007e-02
## Czech Republic         8.333333e-03 2.069461 1.627507e-02
## Egypt                  7.692308e-02 1.603218 1.233245e-01
## England                1.138952e-03 2.927072 3.019323e-03
## France                 1.888324e-05 3.208562 4.312555e-05
## Georgia                1.432665e-03 2.389063 3.076401e-03
## Germany                8.309789e-05 3.027114 2.032708e-04
## Greece                 3.254149e-04 2.650854 7.302661e-04
## Hungary                8.920607e-04 2.383168 1.884902e-03
## India                  1.075269e-02 2.020152 2.110089e-02
## Israel                 2.706360e-04 2.556884 5.429332e-04
## Italy                  1.926819e-05 3.273428 4.528185e-05
## Lebanon                2.557545e-03 2.277371 5.493128e-03
## Luxembourg             1.923077e-02 1.930510 3.349757e-02
## Macedonia              1.098901e-02 2.056275 2.179689e-02
## Mexico                 1.550388e-03 2.342194 3.440945e-03
## Moldova                2.369668e-03 2.071168 4.401542e-03
## Morocco                4.310345e-03 2.083415 7.902308e-03
## New Zealand            1.220405e-04 2.849948 2.803336e-04
## Peru                   5.681818e-03 2.253684 1.230788e-02
## Portugal               6.708258e-05 2.943634 1.464804e-04
## Romania                1.221001e-03 2.249330 2.379608e-03
## Serbia                 1.492537e-02 1.944555 2.815663e-02
## Slovakia               1.000000e-01 2.569685 2.569685e-01
## Slovenia               1.472754e-03 2.224218 3.015303e-03
## South Africa           1.015847e-04 2.780424 2.223285e-04
## Spain                  3.231331e-05 3.086785 7.588531e-05
## Switzerland            2.173913e-02 2.148207 4.477741e-02
## Turkey                 1.371742e-03 2.333853 2.804116e-03
## Ukraine                8.928571e-03 2.051826 1.776187e-02
## Uruguay                9.523810e-04 2.402977 2.109997e-03
## US                     5.727016e-06 3.501175 1.472558e-05
```

Hide

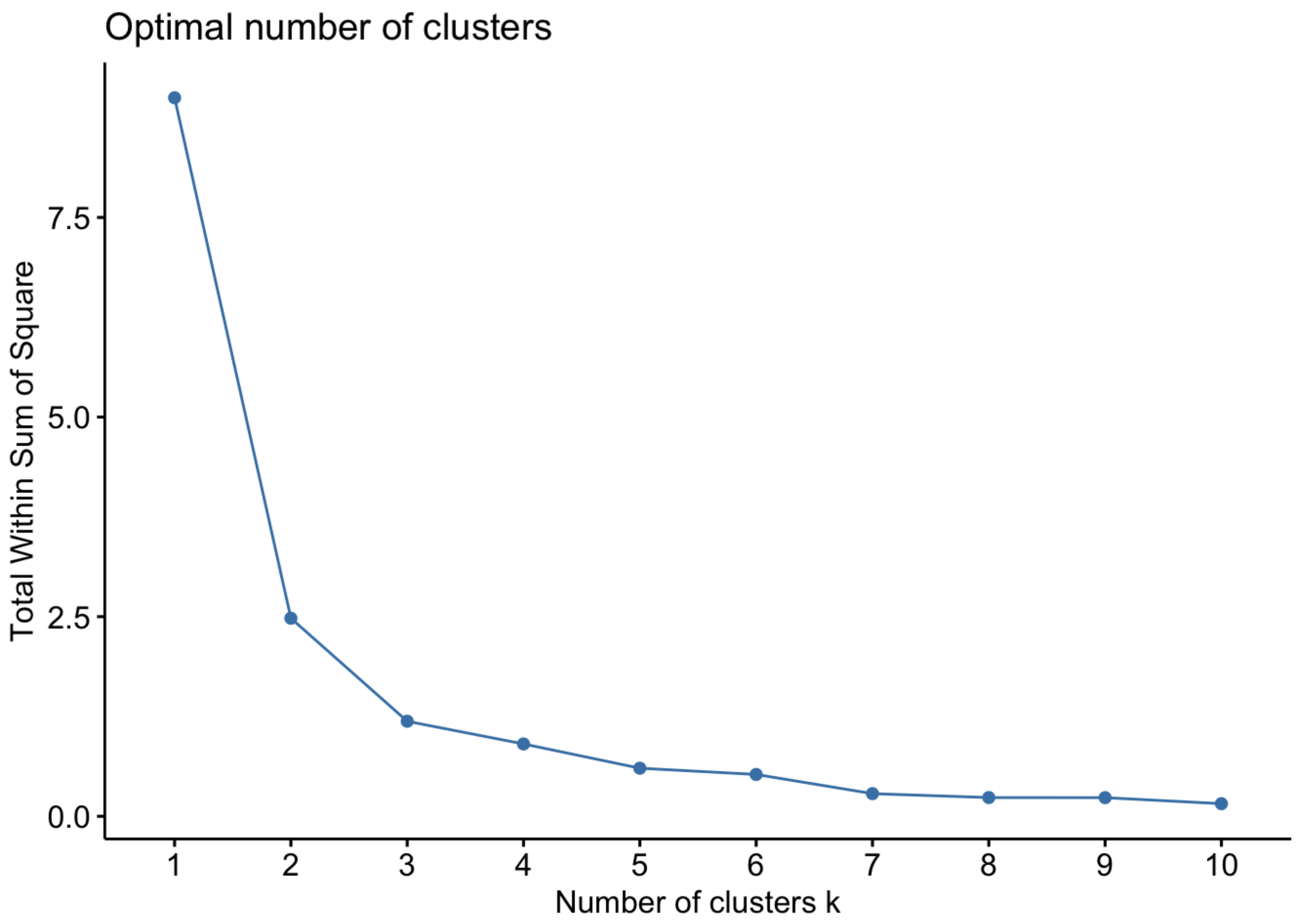
```
tf2<-as.matrix(tf_idf2)
tf2<- na.omit(tf2)
```

Next, we want to determine the optimal number of clusters. One way to do this is by using the `cluster` and `factoextra` package. We can use the fviz_nbclust function to create an elbow plot of number of clusters vs total within sum of squares to determine the optimal number of clusters.

Hide

```
library(cluster)
library(factoextra)

#create plot of number of clusters vs total within sum of squares
fviz_nbclust(tf2, kmeans, method = "wss")
```



Based on this plot, we could say 3 clusters may be optimal. We will now place the data observations into their respected clusters and plot the results.

Hide

```
ktf2<-kmeans(tf2, 3)
```

Hide

```
tf3 <- cbind(tf2, cluster = ktf2$cluster)
tf3
```

```
##                             tf_mean idf_mean   tf_idf_mean cluster
## Argentina              5.106991e-05 2.965020 1.181371e-04       2
## Armenia                7.692308e-02 1.974302 1.518694e-01       3
## Australia              7.089685e-05 2.943949 1.702110e-04       2
## Austria                5.556790e-05 3.076185 1.406401e-04       2
## Bosnia and Herzegovina 4.000000e-02 1.802920 7.211681e-02       3
## Brazil                 2.100840e-03 2.316951 4.579668e-03       1
## Bulgaria               1.049318e-03 2.209334 1.866497e-03       1
## Canada                 4.777831e-04 2.609479 1.135875e-03       1
## Chile                  4.507144e-05 2.979051 1.040230e-04       2
## China                  5.555556e-02 1.691157 9.395314e-02       3
## Croatia                1.351351e-03 2.334815 2.900878e-03       1
## Cyprus                 9.708738e-03 2.527374 2.406007e-02       1
## Czech Republic         8.333333e-03 2.069461 1.627507e-02       3
## Egypt                  7.692308e-02 1.603218 1.233245e-01       3
## England                1.138952e-03 2.927072 3.019323e-03       2
## France                 1.888324e-05 3.208562 4.312555e-05       2
## Georgia                1.432665e-03 2.389063 3.076401e-03       1
## Germany                8.309789e-05 3.027114 2.032708e-04       2
## Greece                 3.254149e-04 2.650854 7.302661e-04       1
## Hungary                8.920607e-04 2.383168 1.884902e-03       1
## India                  1.075269e-02 2.020152 2.110089e-02       3
## Israel                 2.706360e-04 2.556884 5.429332e-04       1
## Italy                  1.926819e-05 3.273428 4.528185e-05       2
## Lebanon                2.557545e-03 2.277371 5.493128e-03       1
## Luxembourg             1.923077e-02 1.930510 3.349757e-02       3
## Macedonia              1.098901e-02 2.056275 2.179689e-02       3
## Mexico                 1.550388e-03 2.342194 3.440945e-03       1
## Moldova                2.369668e-03 2.071168 4.401542e-03       3
## Morocco                4.310345e-03 2.083415 7.902308e-03       3
## New Zealand            1.220405e-04 2.849948 2.803336e-04       2
## Peru                   5.681818e-03 2.253684 1.230788e-02       1
## Portugal               6.708258e-05 2.943634 1.464804e-04       2
## Romania                1.221001e-03 2.249330 2.379608e-03       1
## Serbia                 1.492537e-02 1.944555 2.815663e-02       3
## Slovakia               1.000000e-01 2.569685 2.569685e-01       1
## Slovenia               1.472754e-03 2.224218 3.015303e-03       1
## South Africa           1.015847e-04 2.780424 2.223285e-04       2
## Spain                  3.231331e-05 3.086785 7.588531e-05       2
## Switzerland            2.173913e-02 2.148207 4.477741e-02       3
## Turkey                 1.371742e-03 2.333853 2.804116e-03       1
## Ukraine                8.928571e-03 2.051826 1.776187e-02       3
## Uruguay                9.523810e-04 2.402977 2.109997e-03       1
## US                     5.727016e-06 3.501175 1.472558e-05       2
```

We can identify the means for each cluster to get a sense of the groupings.

Hide

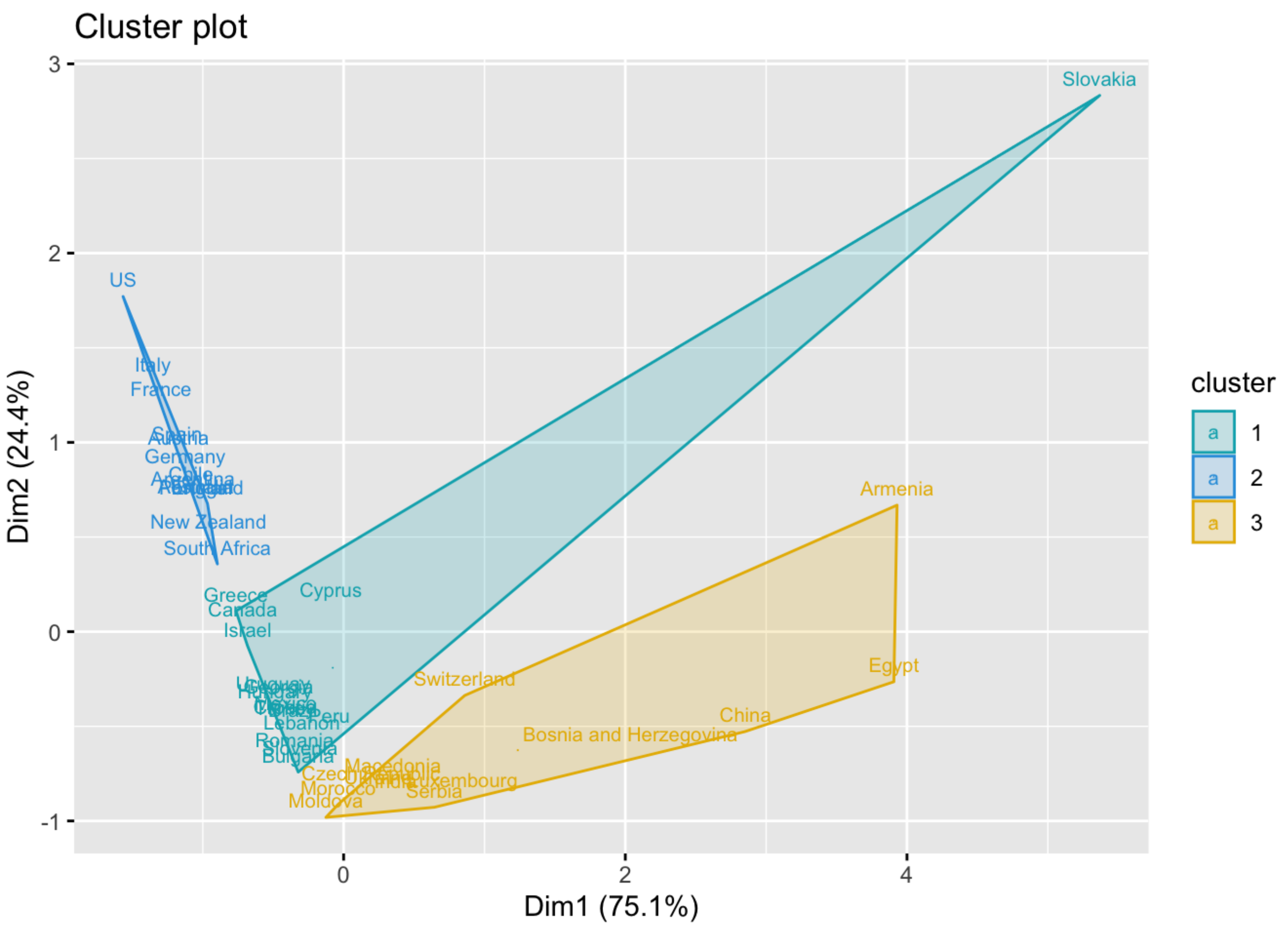
```
agg <-aggregate(tf2, by=list(cluster=ktf2$cluster), mean)
agg
```

```
##   cluster      tf_mean idf_mean  tf_idf_mean
## 1       1 0.0077892021 2.390073 0.0193704089
## 2       2 0.0001393504 3.043257 0.0003525974
## 3       3 0.0269985076 1.957474 0.0489949229
```

We can visualize the different clusters using the function fviz_cluster() [factoextra package]. It can be used to easily visualize k-means clusters. It takes k-means results and the original data as arguments. In the resulting plot, observations are represented by points, using principal components if the number of variables is greater than 2. In order to see the countries for each cluster, we add the country names on the plot for each cluster.

Hide

```
fc<-fviz_cluster(ktf2, data = tf2, geom = "text", labelsize=8,
             palette = c("#00AFBB","#2E9FDF", "#E7B800"))
fc
```



Since we were primarily looking at the United States, Italy, and France in our previous work, we would like to specifically look at these countries to see if they make up different clusters or the same one. Based on the cluster plot above, we can determine that the United States, Italy, and France make up the same cluster. We also see that some countries are clustered based on the region they are from around the world. For example, we see many western European countries in cluster 1. This clustering analysis can be helpful when identifying what countries' wines are more similar and what countries' wines are not. While we have done k-means clustering in previous coursework, it is very helpful to be able to extend this clustering analysis technique to text data.

Summary and Future Steps

After going through this tutorial, we hope you have a better understanding of Natural Language Processing and Exploratory Text Analytics. NLP is a very powerful tool in machine learning as it allows us to explore and gain insight from data that we would not expect to be able to. We demonstrated different visualizations to provide Exploratory Text Analysis that provided insight on the text from this dataset. We also performed Lexicon Sentiment Analysis using the NRC, Bing, and Afinn lexicons and saw what kind of emotions and attitudes were portrayed in these reviews. We then used Term Frequency-Inverse Document Frequency Analysis to transform the textual data into numerical data that we could use to perform k-means clustering to identify what countries were clustered together based on their wine reviews.

While the work we have done already demonstrates the many uses of Natural Language Processing and Text Analytics, there are many other powerful methods and extensions to this area of machine learning. Another interesting technique utilizing text data is using this information to build some kind of prediction model. By using the data we have, we could train a model to predict where a wine is from, for example, based on its review. This is just one example of how machine learning can be used for providing meaningful insights on textual data. While we often think of data as numbers, it is important to remember that data can be extracted from nearly everything, including text.

Sources

https://www.ibm.com/cloud/learn/natural-language-processing (https://www.ibm.com/cloud/learn/natural-language-processing)

https://monkeylearn.com/blog/what-is-tf-idf/ (https://monkeylearn.com/blog/what-is-tf-idf/)

https://afit-r.github.io/sentiment_analysis (https://afit-r.github.io/sentiment_analysis)

https://www.kaggle.com/code/maksymbilyi/k-means-of-text-analyzing-wine-reviews-tf-idf (https://www.kaggle.com/code/maksymbilyi/k-means-of-text-analyzing-wine-reviews-tf-idf)