

will the real shakespeare please stand up?



September 16, 2018

ist 585

LUECK

## Abstract:

While most Shakespeare scholars have little doubt as to the authorship of William Shakespeare’s works, a minority view posits that either Sir Francis Bacon or Edward de Vere are the real authors. Implementing several data mining tools, this paper attempts to examine whether the works attributed to William Shakespeare have any overlap with the other two writers.

## Introduction:

Most Shakespeare scholars belong to what’s known as the Stratfordian school of thought which says that William Shakespeare wrote all his own poems and plays. However, since the mid-nineteenth century, the Baconian theory has argued that Sir Francis Bacon, a distinguished philosopher and scientist wrote all of Shakespeare’s poems and plays under a pen name. Stemming from this idea, a third school of thought came about in the early 1900s. This new idea, known as the Oxfordian theory, thinks that Edward de Vere, the Earl of Oxford during Shakespeare’s lifetime, was the original author. Both Baconians and Oxfordians have similar arguments as to why William Shakespeare couldn’t have written his plays and poems. The main argument is that William Shakespeare’s family didn’t have enough money to provide him with the education necessary to have a full comprehension of the subjects that appear in his works. The proof of these theories relies on ties to similarities in language and phrases used by either Sir Francis Bacon and Edward de Vere as compared to William Shakespeare.

Stratfordians such as Dr. Patrick Cheney of Penn State University dismiss these ideas as conspiracy theories (Stevenson). Although there isn’t much about followers of the Baconian theory anymore, plenty of respected scholars and writers alike cling to the idea that William Shakespeare was not the original author of his works. One major supporter of the Oxfordian theory, The Shakespeare Oxford Fellowship, is a non-profit organization formed in 2014 dedicated to “investigating the Shakespeare authorship question and disseminating the evidence that Edward de Vere, the 17th Earl of Oxford (1550 – 1604), is the true author of the poems and plays published under the pseudonym ‘William Shakespeare’(Shakespeare Oxford Fellowship Board of Trustees).” This organization publishes several newsletters and journals as well as maintains a website dedicated to this debate.

## Method:

This paper explores all three theories using decision trees (DT), k-means, k-nearest neighbor (KNN), and support vector machine (SVM) to view the similarities and differences between the three writers. To normalize the amount of writing per author, only the first 100 sonnets by William Shakespeare are included. The included sonnets were found at gutenberg.org. Sir Francis Bacon’s 19 essays were downloaded from gutenberg.org as well. For Edward de Vere, the data set includes 20 poems from shakespeareoxfordfellowship.org and 15 letters collected from and oxford-shakespeare.com.

Once all works were collected, any English stop words such as “the” or “a”, and Elizabethan stop words such as “thou” or “thy” were removed. Some of the Elizabethan stop words were compiled by Bryan Bumgardner and were copied from his website http://bryanbumgardner.com. Below is a full record of the additional stop words:

art, can, doth, dost, ere, hast, hath, hence, hither, nigh, oft, should’st, thither, tither, thee, thou, thine, thy, tis, t’was, wast, whence, wherefore, whereto, will, withal, would’st, ye, yon, yonder, lordship, come, hamlet, cause, also, use, said, see, other, say, exeunt, ham, man, men, shall

After removing all stop words and cleaning the data, the texts were put into a corpus for data analysis.

## Data Exploration:

Upon completion of the preprocessing stage, each author’s word count was examined to minimize any bias toward one author. The word count per author and per document are as follows:

|  |  |
| --- | --- |
| Author | Word Count |
| Shakespeare | 5438 |
| Bacon | 5348 |
| De Vere | 4556 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Minimum | 1st Quartile | Median | Mean | 3rd Quartile | Maximum |
| 22 | 52 | 58 | 98.98 | 107 | 508 |

The maximum in the above line is an outlier. However, that outlier represents Shakespeare’s play Julius Caesar. Without this outlier, the data would be solely based on Shakespeare’s Sonnets

In addition to viewing the word count, most reviews of these theories emphasize the similarity between words and phrases used by either Bacon or de Vere, and Shakespeare. The top five words across the various collected words were:

**“good”, “love”, “make”, “time”, “yet”**

The following word cloud demonstrates the seventy most frequently used words found in the collected works of these three authors.



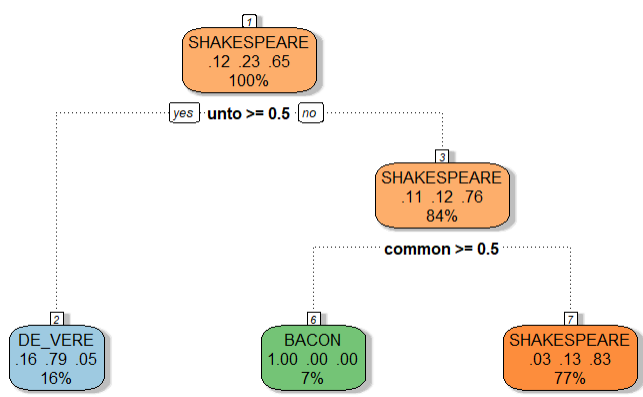
Additionally, the following table illustrates the top five most common word associations for the top three words along with how frequently these words are paired:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time** | | **Love** | | **Good** | |
| rejected | 0.65 | absurdly | 0.68 | otherwise | 0.68 |
| course | 0.60 | admit | 0.68 | thing | 0.61 |
| continuance | 0.58 | alteri | 0.68 | citizen | 0.60 |
| cutting | 0.54 | amorous | 0.68 | things | 0.60 |
| speeches | 0.52 | antonius | 0.68 | affect | 0.55 |

A more comprehensive list of frequently used words can be found by running the code against the included data set. The code appendix also includes a method for viewing the number of words per document.

## Decision Tree Method:

Statisticians and data scientists have typically used a Bayesian approache to identify authorship in documents for natural language processing. One of the most famous instances of one of these methods in use to identify an author can be found in Inference and Disputed Authorship by Frederick Mosteller and David L. Wallace wherein they used Bayesian statistics to identify that Madison authored the disputed Federalist Papers written under the pseudonym “Publius” (Nerbonne 6). This study was able to predict that James Madison was the author of the disputed documents based heavily on his use, or lack thereof, of the word “upon” (Nerbonne 7). The following model attempts to identify authorship based on the word selection per document.



This method does identify the word “unto” as a word frequently used by both Shakespeare and de Vere, and the word “common” as frequently used by Shakespeare and Bacon. However, the probability of choosing the correct author per document using this model only returns the low accuracy level of ~70% with a confidence interval (CI) of between ~52% and ~83%. The Kappa prediction rate is much lower than any of those figures with an accuracy level of ~29% leading to the belief that this model may be just slightly better than randomly guessing which of the three authors wrote each document. The two tables listed below compare Shakespeare with each author individually to test the CI on a smaller scale.

## k-Means Method:

The k-means method was integrated in two ways. The first attempt includes three centers to try to account for the words used by each author. The second way attempts to divide the documents by author also using three centers. Both instances of the k-means test allow for a maximum of 50 iterations with 25 starting points. They also normalize the data by incorporating the tf-idf method.

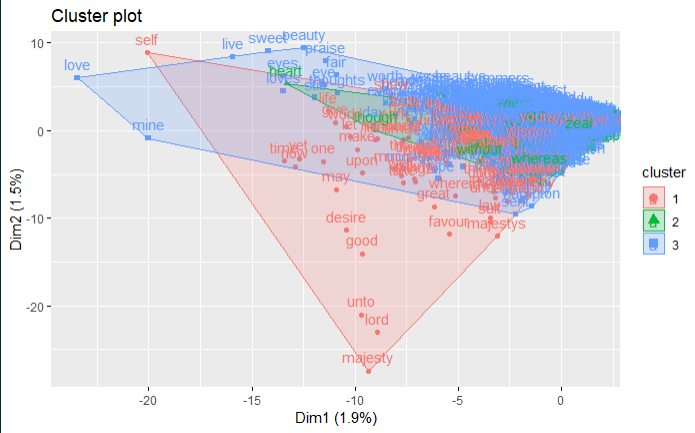


Figure 1 k-means analysis words per author

The above figure shows data from Shakespeare in blue, de Vere in red, and Bacon in green. Attempting to separate the words by author shows that, while very few words stick out, most of the vocabulary found in the corpus overlaps significantly for all three authors.

Additionally, the below figure illustrates that there’s lots of overlap in most of the documents for each author with only a couple of Bacon’s works standing out.

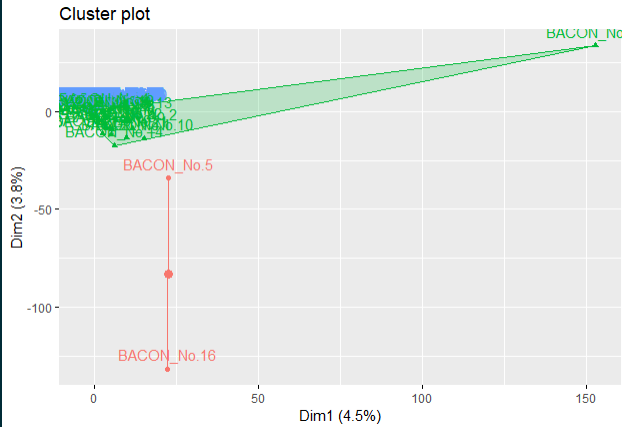
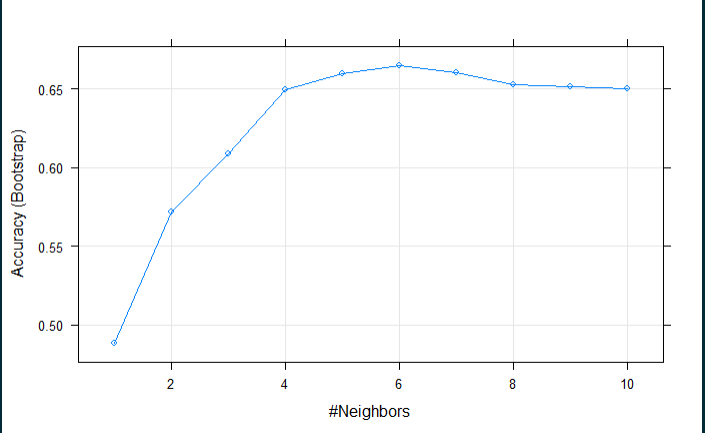


Figure 2 k-means- author per document

## k-Nearest Neighbor (KNN) Method:

Typically, if the k-means model has difficulty in separating data the k-nearest neighbor is often used to split the data further by grouping each data point to its nearest points as specified by the user. Using KNN on this data set results in a reading of ~64% accuracy with a confidence interval between ~47% and ~79%. For the highest accuracy, the model recommends incorporating 6 neighbors when using the bootstrap method according to the plotted data below:



## Support Vector Machine (SVM) Method:

Support vector machines can work with both regression and categorical problems. In the case of categorical data, the support vector machine will try to find a hyperplane of separation between different categories and exploit that point of separation to categorize the data. Since this method is considered an assisted learning algorithm, the user provides some guidance as to how to separate the data. In this model, a cost of 25 was assigned to the model which produces about 100 support vectors on average. The outcome has an average accuracy of ~92% with a confidence interval between ~76% and ~97%. The kappa value is ~79%.

## Conclusion:

While most models don’t find enough information in the given documents to provide a solid confidence interval to debunk any of the given theories, a well-tuned support vector machine provides the most support for Stratfordians. More investigation into how well these authors correlate with Shakespeare’s plays, and further research into how their word associations match from a contextual point of view would help to either prove or disprove these theories. However, given the time constraints and scope of this project, those data will have to be explored at another time.

## References:

Nerbonne, John. “The Exact Analysis of Text.” semanticscholar.org. https://pdfs.semanticscholar.org/4c15/7f110dfd1d90f9931205003719f4c9856d01.pdf

Shakespeare Oxford Fellowship Board of Trustees. “Mission Statement.” Shakespeare Oxford Fellowship. https://shakespeareoxfordfellowship.org/about-us/mission-statement/

Stevenson, Alexa. “Probing Question: Did Shakespeare really write all those plays?” Penn State News. https://news.psu.edu/story/141315/2008/12/01/research/probing-question-did-shakespeare-really-write-all-those-plays (Accessed 8/24/2018)

## Code:

SECTION 1: Data Preparation

### IMPORT THE CORRECT LIBRARIES AND GET THE DOCUMENTS ###  
library(Matrix)  
library(mclust)

## Package 'mclust' version 5.4.1  
## Type 'citation("mclust")' for citing this R package in publications.

library(ggplot2)  
library(tidytext)  
library(corpus)  
library(dplyr)

##   
## 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

library(stringr)  
library(tm)

## Loading required package: NLP

##   
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':  
##   
## annotate

#Import the text files  
sb <- 'C:/Users/Randy/Documents/Syracuse/iSchool/IST565/Homework/Project/Works/shake\_and\_bake.txt'  
sb <- readLines(sb, encoding = "UTF-8") #docs with shakespeare and bacon only  
  
sv <- 'C:/Users/Randy/Documents/Syracuse/iSchool/IST565/Homework/Project/Works/shakes\_vere.txt'  
sv <- readLines(sv, encoding = "UTF-8") #docs with shakespeare and de vere only  
  
svb <- 'C:/Users/Randy/Documents/Syracuse/iSchool/IST565/Homework/Project/Works/all\_works\_sonnets.txt'  
svb <- readLines(svb, encoding = "UTF-8") #docs with all three authors

#extract author/document data for later use  
get\_author <- function(doc\_list){  
 author <- grep('SHAKESPEARE\_No.|DE\_VERE\_No.|BACON\_No.', doc\_list, value = TRUE)  
 author <- gsub(":", "", author)  
 author <- trimws(author)  
 return(author)  
}

### SEPARATE THE DOCUMENTS PER AUTHOR###  
  
sep\_by\_author <- function(raw\_text){  
 #define the title  
 author\_title <- grep("SHAKESPEARE\_No.|DE\_VERE\_No.|BACON\_No.", raw\_text)  
   
 #define document body per doc using title and end\_doc  
 doc\_end <- c(author\_title -1, length(raw\_text))  
 doc\_end <- doc\_end[-1]  
   
 #separate the documents using the start and end index  
 text <- mapply(function(db, de) paste(raw\_text[db:de], collapse = "\n"), author\_title, doc\_end)  
   
 #assign the output to the new variable  
 return(text)  
}

### CLEAN UP THE TEXT ###  
text\_clean <- function(doc\_list){  
 #replace newline characters with spaces  
 doc\_list <- gsub("[\r\n]", " ", doc\_list)  
   
 #set all words to lowercase  
 doc\_list <- tolower(doc\_list)  
   
 #remove symbols that didn't convert to utf-8 properly  
 doc\_list <- iconv(doc\_list, "latin1", "ASCII", sub="")  
   
 #additional stop words  
 addl\_words <- c('art', 'can',  
 'doth', 'dost', '\'ere', 'hast', 'hath', 'hence', 'hither', 'nigh', 'oft', 'should\'st', 'thither', 'tither',  
 'thee', 'thou', 'thine', 'thy', '\'tis', '\'twas', 'wast', 'whence', 'wherefore', 'whereto', 'will', 'withal',  
 'would\'st', 'ye', 'yon', 'yonder', 'lordship', 'come', 'hamlet', 'cause', 'also', 'use', 'said', 'see','other',  
 'say', 'exeunt', 'ham', 'man', 'men', 'shall')  
 #remove all stop words  
 doc\_list <- removeWords(doc\_list, c(stopwords("english"), addl\_words))  
   
 #retain Elizabethan stemmed words  
 doc\_list <- gsub("\'d", "ed", doc\_list)  
   
 #remove titles that include author names  
 doc\_list <- gsub("shakespeare\_no.", "", doc\_list)  
 doc\_list <- gsub("de\_vere\_no.", "", doc\_list)  
 doc\_list <- gsub("bacon\_no.", "", doc\_list)  
   
 #remove whitespace  
 doc\_list <- trimws(doc\_list)  
   
 return(doc\_list)  
}

### RUN THE ABOVE FUNCTIONS ON THE DATA ###  
  
sb\_authors <- get\_author(sb)  
sv\_authors <- get\_author(sv)  
svb\_authors <- get\_author(svb)  
  
sb <- sep\_by\_author(sb)  
sv <- sep\_by\_author(sv)  
svb <- sep\_by\_author(svb)  
  
sb <- text\_clean(sb)  
sv <- text\_clean(sv)  
svb <- text\_clean(svb)

SECTION 2: Set up data for exploration

### CREATE THE CORPUS FOR THE CHOSEN AUTHORS###  
ws\_corpus <- VCorpus(VectorSource(svb))

### NORMALIZE THE CORPUS ###  
  
#var to capture the number of documents  
(numdocs <- length(ws\_corpus))

## [1] 155

#minimum and maximum term frequencies  
minFreq <- numdocs \* 0.001  
maxFreq <- numdocs \* 1

### CREATE A TERM DOCUMENT MATRIX ###  
ws\_dtm <- DocumentTermMatrix(ws\_corpus,  
 control = list(  
 removePunctuation = T,  
 wordLengths = c(3,15),  
 stemming = T,  
 removeNumbers = T,  
 remove\_separators = T,  
 bounds = list(global = c(minFreq, maxFreq))))  
#turn the document term matrix into a matrix  
ws\_matrix <- as.matrix(ws\_dtm)  
  
#provide row names to match documents per row (ENTER THE CORRECT AUTHOR LIST)  
row.names(ws\_matrix) <- svb\_authors

SECTION 3: Data Discovery

### DATA DISCOVERY ###  
  
#view number of words per author  
  
#tdm data frame (ENTER THE CORRECT AUTHOR LIST))  
ws\_df <- data.frame(ws\_matrix)  
ws\_df$author <- gsub("\_No.\*", "", svb\_authors)  
  
#get the author column name  
author\_col <- match('author', names(ws\_df))  
  
#Rearrange the columns to put author in the first row  
ws\_df <- subset(ws\_df, select=c(author\_col,1:(author\_col - 1),(author\_col + 1):length(ws\_df)))  
  
#Add a totals column for row totals  
ws\_df$totals <- rowSums(ws\_df[,2:length(ws\_df)])  
  
#summarize word count per author  
author\_words <- ws\_df %>%  
 group\_by(author) %>%  
 summarise(sum(totals))  
  
colnames(author\_words) <- c('author', 'wordcount')  
  
author\_words

## # A tibble: 3 x 2  
## author wordcount  
## <chr> <dbl>  
## 1 BACON 5348  
## 2 DE\_VERE 4556  
## 3 SHAKESPEARE 5438

#summarize words per piece  
piece\_words <- ws\_df %>%  
 group\_by(row.names(ws\_df)) %>%  
 summarise(sum(totals))  
  
piece\_words

## # A tibble: 155 x 2  
## `row.names(ws\_df)` `sum(totals)`  
## <chr> <dbl>  
## 1 BACON\_No.1 357  
## 2 BACON\_No.10 370  
## 3 BACON\_No.11 217  
## 4 BACON\_No.12 426  
## 5 BACON\_No.13 221  
## 6 BACON\_No.14 340  
## 7 BACON\_No.15 139  
## 8 BACON\_No.16 508  
## 9 BACON\_No.17 214  
## 10 BACON\_No.18 177  
## # ... with 145 more rows

#get the average word count per document  
summary(piece\_words$'sum(totals)')

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 22.00 52.00 58.00 98.98 107.00 508.00

#view words with a frequency >= 10  
(top\_words <- colnames(ws\_matrix[, which(colSums(ws\_matrix)>=100)]))

## [1] "good" "love" "make" "time" "yet"

#Word cloud setup  
  
#transform the matrix in into a data frame  
library(wordcloud)

## Loading required package: RColorBrewer

wc\_tdm <- TermDocumentMatrix(ws\_corpus,  
 control = list(  
 removePunctuation = T,  
 wordLengths = c(3,18),  
 #stemming = T,  
 removeNumbers = T,  
 remove\_separators = T,  
 bounds = list(global = c(minFreq, maxFreq))  
 ))  
wc\_m <- as.matrix(wc\_tdm)  
wc\_filter <- sort(rowSums(wc\_m),decreasing=TRUE)  
wc\_df <- data.frame(word = names(wc\_filter),freq=wc\_filter)  
  
set.seed(177)  
wordcloud(words = wc\_df$word, freq = wc\_df$freq, min.freq = 3,  
 max.words=70, random.order=FALSE, rot.per=0.35,   
 colors=brewer.pal(8, "Dark2"))



#view some word associations for the top three words  
(time\_assoc <- data.frame(findAssocs(wc\_tdm, terms = "time", corlimit = 0.4)))

## time  
## rejected 0.61  
## continuance 0.54  
## speeches 0.54  
## cutting 0.51  
## passages 0.51

(love\_assoc <- data.frame(findAssocs(wc\_tdm, terms = "love", corlimit = 0.4)))

## love  
## absurdly 0.68  
## admit 0.68  
## alteri 0.68  
## amorous 0.68  
## antonius 0.68

(good\_assoc <- data.frame(findAssocs(wc\_tdm, terms = "good", corlimit = 0.4)))

## good  
## otherwise 0.65  
## thing 0.59  
## citizen 0.58  
## things 0.57  
## affect 0.54

### BUILD A TRAINING AND TEST SET ###  
library(caret)

## Loading required package: lattice

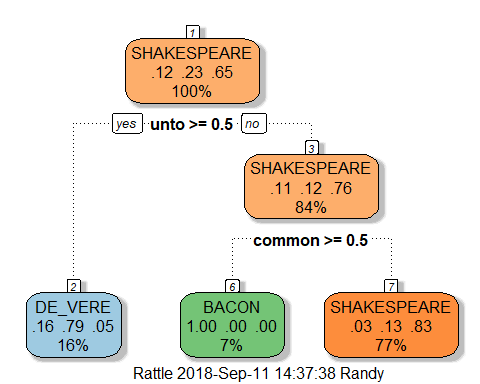
library(rpart)  
library(e1071)  
  
#function to pull a data sample given a data frame  
sample\_index <- function(df){  
 df\_index <- seq(1,nrow(df),4)  
 return(df\_index)  
}  
  
#make a test indexer  
test\_index <- sample\_index(ws\_df)  
  
#assign the test indexer rows to test and everything else to train  
ws\_train <-ws\_df[-test\_index,]  
ws\_test <- ws\_df[test\_index,]  
  
#remove totals row from data frames  
ws\_train <- subset(ws\_train, select = -c(totals))  
ws\_test <- subset(ws\_test, select = -c(totals))  
  
#remove author from the test set  
ws\_test\_noauthor <- subset(ws\_test, select = -c(author))

Build a Decision Tree Model

library(rattle)

## Rattle: A free graphical interface for data science with R.  
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

#train the machine to try predicting the document based on words used by each author  
dt\_model <- train(author ~.,  
 data = ws\_train,  
 metric = 'Accuracy',  
 method = 'rpart')  
   
#view the decision tree  
fancyRpartPlot(dt\_model$finalModel)



Use the Decision Tree Model to Identify Authors from the Test Set

dt\_predict <- predict(dt\_model, ws\_test\_noauthor, type = 'raw')  
dt\_confusionMatrix <- confusionMatrix(dt\_predict, as.factor(ws\_test$author))  
dt\_confusionMatrix

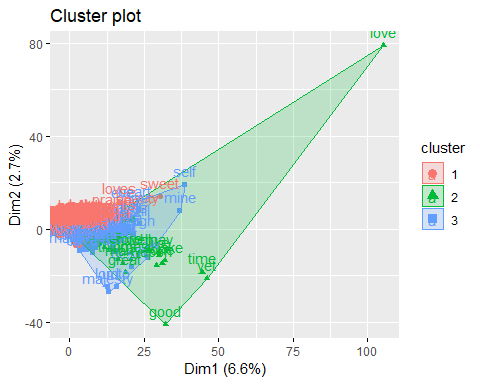
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction BACON DE\_VERE SHAKESPEARE  
## BACON 2 0 2  
## DE\_VERE 0 2 0  
## SHAKESPEARE 3 7 23  
##   
## Overall Statistics  
##   
## Accuracy : 0.6923   
## 95% CI : (0.5243, 0.8298)  
## No Information Rate : 0.641   
## P-Value [Acc > NIR] : 0.3128   
##   
## Kappa : 0.2888   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: BACON Class: DE\_VERE Class: SHAKESPEARE  
## Sensitivity 0.40000 0.22222 0.9200  
## Specificity 0.94118 1.00000 0.2857  
## Pos Pred Value 0.50000 1.00000 0.6970  
## Neg Pred Value 0.91429 0.81081 0.6667  
## Prevalence 0.12821 0.23077 0.6410  
## Detection Rate 0.05128 0.05128 0.5897  
## Detection Prevalence 0.10256 0.05128 0.8462  
## Balanced Accuracy 0.67059 0.61111 0.6029

Train on the K Means Algorithm to Attempt ML Separation by Author

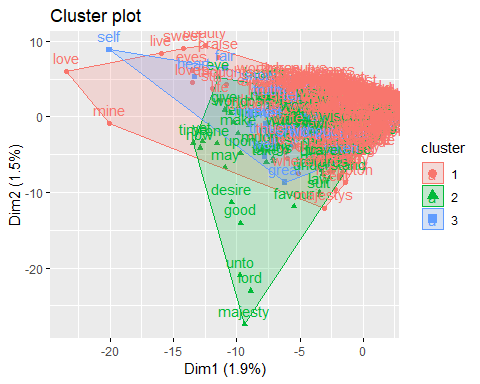
#  
library(factoextra)

## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ

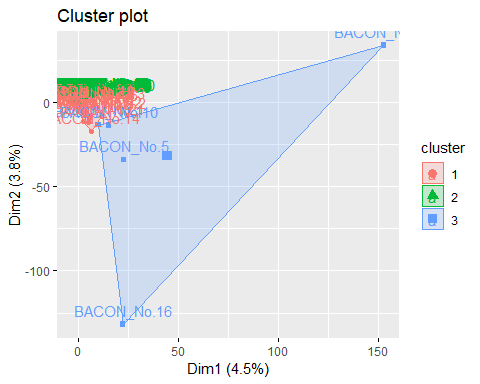
#create wc\_weighted dtm to plot with k-means by word clustering  
wc\_weighted\_tdm <- weightTfIdf(wc\_tdm, normalize = "False")  
wc\_weighted\_tdm <- as.matrix(wc\_weighted\_tdm)  
  
#create ws\_weighted dtm to plot with k-means by author  
ws\_weighted\_dtm <- weightTfIdf(ws\_dtm, normalize = "False")  
ws\_weighted\_dtm <- as.matrix(ws\_weighted\_dtm)  
  
#the first k-means separator uses the wordcloud matrix to   
k1 <- kmeans(wc\_m, centers = 3, iter.max = 50, nstart = 3) #kmeans by word with no weighting  
k2 <- kmeans(wc\_weighted\_tdm, centers = 3, iter.max = 50, nstart = 3) #kmeans by word with tf-idf weighting  
  
k3 <- kmeans(ws\_matrix, centers = 3, iter.max = 50, nstart = 3) #kmeans by author with no weighting  
k4 <- kmeans(ws\_weighted\_dtm, centers = 3, iter.max = 50, nstart = 3) #kmeans by author with weighting  
  
fviz\_cluster(k1, wc\_m)



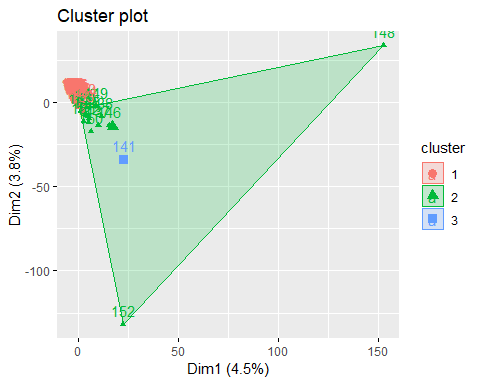
fviz\_cluster(k2, wc\_weighted\_tdm)



fviz\_cluster(k3, ws\_matrix)



fviz\_cluster(k4, ws\_weighted\_dtm)



SVM Model

#set train and test data authors to factors for nb compliance  
svm\_train <- ws\_train  
svm\_train$author <- as.factor(svm\_train$author)  
  
svm\_test <- ws\_test  
svm\_test$author <- as.factor(svm\_test$author)  
  
svm\_test\_noauthor <- subset(svm\_test, select = -c(author))  
  
model\_svm <- svm(author ~.,  
 data = svm\_train,  
 kernel = "radial",  
 cost = 25,  
 scale = FALSE)  
  
svm\_prediction <- predict(model\_svm, svm\_test\_noauthor)  
svm\_confusionMatrix <- confusionMatrix(svm\_prediction, svm\_test$author)  
svm\_confusionMatrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction BACON DE\_VERE SHAKESPEARE  
## BACON 5 0 0  
## DE\_VERE 0 5 0  
## SHAKESPEARE 0 4 25  
##   
## Overall Statistics  
##   
## Accuracy : 0.8974   
## 95% CI : (0.7578, 0.9713)  
## No Information Rate : 0.641   
## P-Value [Acc > NIR] : 0.0002924   
##   
## Kappa : 0.7851   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: BACON Class: DE\_VERE Class: SHAKESPEARE  
## Sensitivity 1.0000 0.5556 1.0000  
## Specificity 1.0000 1.0000 0.7143  
## Pos Pred Value 1.0000 1.0000 0.8621  
## Neg Pred Value 1.0000 0.8824 1.0000  
## Prevalence 0.1282 0.2308 0.6410  
## Detection Rate 0.1282 0.1282 0.6410  
## Detection Prevalence 0.1282 0.1282 0.7436  
## Balanced Accuracy 1.0000 0.7778 0.8571

print(model\_svm)

##   
## Call:  
## svm(formula = author ~ ., data = svm\_train, kernel = "radial",   
## cost = 25, scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 25   
## gamma: 0.000269324   
##   
## Number of Support Vectors: 102

KNN Model

model\_knn <- train(author ~., data = svm\_train,  
 method = "knn",  
 tuneGrid = data.frame(k = seq(1,10)),  
 trControl = trainControl(method = 'cv'))  
  
knn\_prediction <- predict(model\_knn, svm\_test\_noauthor)  
knn\_confusionMatrix <- confusionMatrix(knn\_prediction, svm\_test$author)  
knn\_confusionMatrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction BACON DE\_VERE SHAKESPEARE  
## BACON 0 0 0  
## DE\_VERE 2 0 0  
## SHAKESPEARE 3 9 25  
##   
## Overall Statistics  
##   
## Accuracy : 0.641   
## 95% CI : (0.4718, 0.788)  
## No Information Rate : 0.641   
## P-Value [Acc > NIR] : 0.572191   
##   
## Kappa : 0.0554   
## Mcnemar's Test P-Value : 0.002905   
##   
## Statistics by Class:  
##   
## Class: BACON Class: DE\_VERE Class: SHAKESPEARE  
## Sensitivity 0.0000 0.00000 1.0000  
## Specificity 1.0000 0.93333 0.1429  
## Pos Pred Value NaN 0.00000 0.6757  
## Neg Pred Value 0.8718 0.75676 1.0000  
## Prevalence 0.1282 0.23077 0.6410  
## Detection Rate 0.0000 0.00000 0.6410  
## Detection Prevalence 0.0000 0.05128 0.9487  
## Balanced Accuracy 0.5000 0.46667 0.5714

#graph the best amount of neighbors for prediction  
plot(model\_knn)

