Parin Patel

Use Clustering to Solve a Mystery in History

Introduction:

In the late 1780's, to address the U.S Constitution still needing approval from nine of the thirteen states, The Federalist Papers, a series of 85-letters, were written in newspapers urging the ratification of the U.S Constitution. These letters contended that the US Constitution would preserve the Union and allow for the proposed federal government to act in the nations interest. The Federalist Papers have long been regarded as an extremely important source of evidence of the underlying meaning and intentions of the Constitution, as well. In fact, in 1821, in *Cohens v Virginia* Chief Justice John Marshall described the papers as the following:

> "complete commentary on our constitution; and its appealed to by all parties in the questions to which that instrument has given birth. Its intrinsic merit entitles it to this high rank, and the part two of its authors performed in framing the constitution, put it very much in their power to explain the views with which it was framed."

The predicament is, for a document with this caliber of importance in creating and ratifying our country's' Constitution, we do not know completely who wrote all the essays that make up the Federalist Papers. In fact, 11 essays are still disputed about regarding whether Madison or Hamilton wrote them. By using the K-Means, EM, and HAC clustering algorithms, we hope to draw a conclusion on who wrote the disputed essays by demonstrating what patterns were learned to predict the disputed papers.

Analysis:

Data Preparation:

Data Preparation first required loading the .csv file. A general assessment of the file was done using str(data) (Figure 1) . We then looked for complete cases by returning a logical vector that indicated which cases are complete. It showed that we have no missing value (Figure 2)

Parin Patel

```
> str(data)
'data.frame':	85 obs. of  72 variables:
 $ author  : Factor w/ 5 levels "dispt","Hamilton",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ filename: Factor w/ 85 levels "dispt_fed_49.txt",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ a       : num  0.28 0.177 0.339 0.27 0.303 0.245 0.349 0.414 0.248 0.442 ...
 $ all     : num  0.052 0.063 0.09 0.024 0.054 0.059 0.036 0.083 0.04 0.062 ...
 $ also    : num  0.009 0.013 0.008 0.016 0.027 0.007 0.007 0.009 0.007 0.006 ...
 $ an      : num  0.096 0.038 0.03 0.024 0.034 0.067 0.029 0.018 0.04 0.075 ...
 $ and     : num  0.358 0.393 0.301 0.262 0.404 0.282 0.335 0.478 0.356 0.423 ...
 $ any     : num  0.026 0.063 0.008 0.056 0.04 0.052 0.058 0.046 0.034 0.037 ...
 $ are     : num  0.131 0.051 0.068 0.064 0.128 0.111 0.087 0.11 0.154 0.093 ...
 $ as      : num  0.122 0.139 0.203 0.111 0.148 0.252 0.073 0.074 0.161 0.1 ...
 $ at      : num  0.017 0.114 0.023 0.056 0.013 0.015 0.116 0.037 0.047 0.031 ...
 $ be      : num  0.411 0.393 0.474 0.365 0.344 0.297 0.378 0.331 0.289 0.379 ...
 $ been    : num  0.026 0.165 0.015 0.127 0.047 0.03 0.044 0.046 0.027 0.025 ...
 $ but     : num  0.009 0 0.038 0.032 0.061 0.037 0.007 0.055 0.027 0.037 ...
 $ by      : num  0.14 0.139 0.173 0.167 0.209 0.186 0.102 0.092 0.168 0.174 ...
 $ can     : num  0.035 0 0.023 0.056 0.088 0 0.058 0.037 0.047 0.056 ...
```

Figure 1

```
> View(data)
> sum(!complete.cases(data))
[1] 0
```

Figure 2

We then removed the first two columns (author and filename) in order to leave the data frame as just function words and feature values (Figure 3).

```
> fp<-data[,c(-1,-2)]
> fp
      a   all  also    an   and   any   are    as    at    be  been   but    by
o
1  0.280 0.052 0.009 0.096 0.358 0.026 0.131 0.122 0.017 0.411 0.026 0.009 0.140
6
2  0.177 0.063 0.013 0.038 0.393 0.063 0.051 0.139 0.114 0.393 0.165 0.000 0.139
3
3  0.339 0.090 0.008 0.030 0.301 0.008 0.068 0.203 0.023 0.474 0.015 0.038 0.173
0
4  0.270 0.024 0.016 0.024 0.262 0.056 0.064 0.111 0.056 0.365 0.127 0.032 0.167
0
5  0.303 0.054 0.027 0.034 0.404 0.040 0.128 0.148 0.013 0.344 0.047 0.061 0.209
0
6  0.245 0.059 0.007 0.067 0.282 0.052 0.111 0.252 0.015 0.297 0.030 0.037 0.186
0
7  0.349 0.036 0.007 0.029 0.335 0.058 0.087 0.073 0.116 0.378 0.044 0.007 0.102
```

Figure 3

Processing:

The first step of processing required scaling the data in order to standardize the range of features in the data (Figure 4) . By scaling, we can account for any wide range values. We also started preparing for cluster analysis by using the elbow graphing method to figure out the most optimal number of clusters with totalwithinss. We chose to set our seed to 10 and created a function to return the optimal number of clusters. We chose our nstart to be 25, meaning it will generate 25 initial configurations. And then plotted the total within-clusters sum of squares (will be discussed in results).

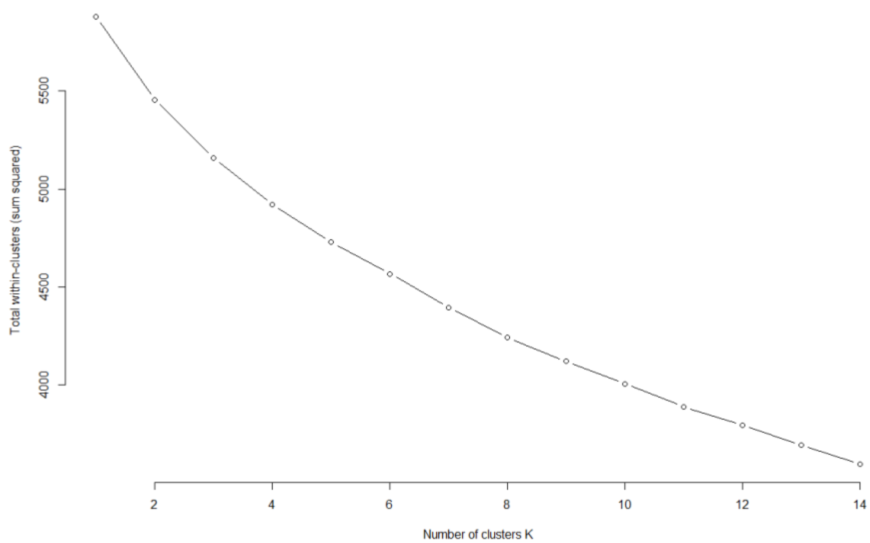


Figure 4

There were two ways of processing the data for K_means. The first involved computing the K-means clustering in R without specifying the Hartigan-Wong method. The second involves using the Hartingan-Wong method to find the centroid.

In the first method, we run the code shown in figure 5. The cluster is set to four and nstart is at 25. The output is shown in results after it was specified to show centers.

```
#### Option 1: K means_Clustering ####
k_mean<-kmeans(fp2, centers = 4, nstart = 25)
k_mean
k_mean$centers
```

Figure 5

In the second method, we used the following two references to better understand the Hartigan-Wong method: https://rstudio-pubs-static.s3.amazonaws.com/79267_cf36e5130fa449bb876ee563908c7a27.html and https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/K-Means

The following line was executed to find the centroid. The results will be discussed later. But, we set our cluster to 4, nstart was 25, and max iterations was set to 100. We specified the algorithm was Hartigan-Wong, and then filtered for results with the center value.

```
######Option 2: :K_means using Hartingan-Wong Method####

##HW algorithm to find centroid
km_output1<-kmeans(fp2,centers=4,nstart = 25,iter.max = 100,algorithm = "Hartigan-Wong")
km_output1$centers
```

Next, we used the HAC clustering algorithm noted below and plotted the results. HAC stands for Hierarchical Cluster Analysis and it is an alternative to K-means clustering for identifying groups in the dataset. HAC does not require a pre-specification of clusters, like K_means did. Additionally, the HAC output is a dendrogram which displays the groups in a tree-shaped plot. The plot can be found in the results section. We set our distance measure to Euclidean and method as complete linkage clustering.

```
# 2nd Algorithm: HAC
hac_output<-hclust(dist(fp2,method = "euclidean"),method = "complete")

#plot HAC
plot(hac_output)

#output desirable number of clusters after modeling
hac_cut<-cutree(hac_output,4)
hac_df1<-data.frame(author=data$author,cluster=hac_cut)
clusplot(fp2,hac_cut,color=TRUE,shade=T,labels = 2,lines =0 )
```

For the third algorithm, we used EM, or Expectation-Maximizing. It is an unsupervised learning algorithm that iteratively finds the maximum likelihood of parameters in a model that depends on unobserved latent variables. We will use the mclust function in R to provide the parameters.

```
#3rd Algorithm: EM
install.packages("mclust")
library(mclust)

#visualize clusters
clPairs(fp2,data$author)
fit<-Mclust(fp2)
summary(fit)
```

Results:

K_Means

Below, the results for both option 1 and 2 are shown (options discussed above). The non-filtered results of both options can be seen in Appendix A. This result was shown to find the clustering sizes. Additionally, it was used to confirm both methods were identical in results. The results shown in Appendix A will confirm this. The four clusters were found to have the following cluster sizes: 4,11,24,46.

The results below show the K_means for when the results are filter for center. In addition, we plotted the centers to better visually analyze the data (Figure 8 & 9).

Option 1 Results:

```
> km_output1$centers
            a          all        also         an        and         any
1  0.28962572 -0.008668348 -0.33688781  0.4564520 -0.4292426  0.31853756
2 -0.57598787  0.176223940 -0.07835166 -0.3045733  0.7401337 -0.88866997
3 -0.06274336  0.038704552  0.41717596 -0.5024666  0.1495853 -0.21337796
4 -1.37026902 -0.617157145  1.58662115 -1.3968214  2.0034108  0.06092825
         are         as         at         be        been         but
1 -0.1269924 -0.1157504  0.1638521  0.1704703  0.07424644 -0.04491703
2 -0.2474613 -0.3541919  0.5272837 -1.3393966  0.27336121 -0.64897176
3  0.4019137  0.3655169 -0.3934412  0.2648322 -0.05293925  0.29481729
4 -0.2705511  0.1120563 -0.9736815  0.1339390 -1.28794190  0.53231438
          by         can          do       down        even      every
1 -0.4488847  0.15202297  0.11283630  0.1846824  0.16870081 -0.1429159
2  0.4744678 -0.88893879 -0.67604087 -0.4239542 -0.39970839 -0.6189158
3  0.6611626  0.07030163 -0.02037322 -0.0890032 -0.07851415  0.6725072
4 -0.1095873  0.27450777  0.68373426 -0.4239542 -0.36977631 -0.6894921
         for.        from        had         has       have         her
1  0.02154164  0.071958773 -0.2033678  0.16005265  0.0596352 -0.19775612
2 -0.06056271  0.008195461  1.6808436  0.01947992  0.4388495  0.02101943
3  0.06123478 -0.286643438 -0.2738688 -0.16743504 -0.1427725 -0.00244272
4 -0.44859010  0.869797219 -0.6403774 -0.88956500 -1.0360062  2.23104823
         his         if.        in.         into         is         it
1  0.1014028  0.09127669  0.4786243 -0.284284711  0.1789476  0.2045855
2  0.2055717 -0.41580531 -0.5515928  0.392435886 -0.8577637 -0.7161222
3 -0.2136100 -0.27657201 -0.4937745 -0.009233076  0.2396741 -0.1122614
4 -0.4497950  1.75321470 -1.0246517  2.245473953 -1.1370926  0.2901717
         its         may       more        must          my          no
1  0.2352606  0.06920864 -0.2880071  0.09850013  0.15721793  0.06580667
2 -0.3206233 -1.11663332 -0.1067859 -0.77537462 -0.02090334 -0.77992250
3 -0.2473462  0.31909596  0.1948770  0.25975274 -0.22413956  0.32764604
4 -0.3397059  0.36026651  2.4364815 -0.55898773 -0.40568466 -0.57786607
          not        now         of         on         one        only
1 -0.04176153  0.1049530  0.3506382 -0.5643201 -0.31726578 -0.07651245
2 -0.58955133  0.3780618 -0.2435579  0.2131969  0.09429220 -0.62459918
3  0.29793854 -0.2331719 -0.2382584  1.0056141  0.07239926  0.38610287
4  0.31389252 -0.8475975 -1.9330044 -0.1302955  2.95485737  0.28092370
           or         our      shall     should          so        some
1  0.05350945  0.01390439  0.1675906  0.2178890 -0.05937977 -0.2618477
2 -0.38365788 -0.26303984 -0.3792315 -0.2724530 -0.08521280  0.6392920
3 -0.19677510 -0.08883363 -0.2231519 -0.2394768  0.06804264  0.2661959
4  1.62035105  1.09646083  0.4545054 -0.3196166  0.50894675 -0.3439797
         such        than          that        the        their
1  0.07150055 -0.02059864  0.2084314394  0.1159819 -0.273306977
2 -0.52709523 -0.50106422 -0.5238465684 -0.4025561  0.991849610
3 -0.02595886 -0.09047437 -0.1592967827  0.3611036  0.008301811
4  0.78300868  2.15765712 -0.0006027944 -2.3933839  0.365632943
         then      there      things       this         to          up
1 -0.07793873  0.4669874  0.20187475  0.3149081  0.6179541  0.16561596
2 -0.04346848 -0.6560934 -0.42002725 -0.3731165 -0.9652637  0.03601269
3  0.12357132 -0.6633081 -0.09329193 -0.1353435 -0.7179521 -0.34049549
4  0.27440577  0.4137506 -0.60673314 -1.7833116 -0.1442837  0.03935447
        upon         was        were       what        when       which
1  0.6879288 -0.23977418 -0.22843317  0.1146205  0.05561194  0.03705055
2 -0.3728410  1.38567795  1.19163527 -0.3776256 -0.24797150 -0.15054096
3 -1.0156057 -0.08612244 -0.11254063 -0.1307965 -0.13136922  0.15286819
4 -0.7922336 -0.53647667  0.02522827  0.5051132  0.83059962 -0.92930284
          who        will        with      would        your
1  0.03114336 -0.05828478 -0.11207025  0.2473179  0.02897163
2  0.27465588 -0.84541229  0.17875034 -0.3482766 -0.20876463
3 -0.16193522  0.37885591 -0.09147142 -0.5037495 -0.06260916
4 -0.14184107  0.72202330  1.34607296  1.1361017  0.61658391
> k_mean$centers
            a          all        also         an        and         any
1 -0.06274336  0.038704552  0.41717596 -0.5024666  0.1495853 -0.21337796
```

Figure 6

Option 2 Results:

```
> ##HW algorithm to find centroid
> km_output1<-kmeans(fp2,centers=4,nstart = 25,iter.max = 100,algorithm = "Ha
gan-Wong")
> km_output1$centers
            a          all        also         an        and         any
1  0.28962572 -0.008668348 -0.33688781  0.4564520 -0.4292426  0.31853756
2 -0.57598787  0.176223940 -0.07835166 -0.3045733  0.7401337 -0.88866997
3 -0.06274336  0.038704552  0.41717596 -0.5024666  0.1495853 -0.21337796
4 -1.37026902 -0.617157145  1.58662115 -1.3968214  2.0034108  0.06092825
         are         as         at         be        been         but
1 -0.1269924 -0.1157504  0.1638521  0.1704703  0.07424644 -0.04491703
2 -0.2474613 -0.3541919  0.5272837 -1.3393966  0.27336121 -0.64897176
3  0.4019137  0.3655169 -0.3934412  0.2648322 -0.05293925  0.29481729
4 -0.2705511  0.1120563 -0.9736815  0.1339390 -1.28794190  0.53231438
          by         can          do       down        even      every
1 -0.4488847  0.15202297  0.11283630  0.1846824  0.16870081 -0.1429159
2  0.4744678 -0.88893879 -0.67604087 -0.4239542 -0.39970839 -0.6189158
3  0.6611626  0.07030163 -0.02037322 -0.0890032 -0.07851415  0.6725072
4 -0.1095873  0.27450777  0.68373426 -0.4239542 -0.36977631 -0.6894921
         for.         from        had         has       have         her
1  0.02154164  0.071958773 -0.2033678  0.16005265  0.0596352 -0.19775612
2 -0.06056271  0.008195461  1.6808436  0.01947992  0.4388495  0.02101943
3  0.06123478 -0.286643438 -0.2738688 -0.16743504 -0.1427725 -0.00244272
4 -0.44859010  0.869797219 -0.6403774 -0.88956500 -1.0360062  2.23104823
         his         if.         in.         into         is          it
1  0.1014028  0.09127669  0.4786243 -0.284284711  0.1789476  0.2045855
2  0.2055717 -0.41580531 -0.5515928  0.392435886 -0.8577637 -0.7161222
3 -0.2136100 -0.27657201 -0.4937745 -0.009233076  0.2396741 -0.1122614
4 -0.4497950  1.75321470 -1.0246517  2.245473953 -1.1370926  0.2901717
         its         may       more        must          my          no
1  0.2352606  0.06920864 -0.2880071  0.09850013  0.15721793  0.06580667
2 -0.3206233 -1.11663332 -0.1067859 -0.77537462 -0.02090334 -0.77992250
3 -0.2473462  0.31909596  0.1948770  0.25975274 -0.22413956  0.32764604
4 -0.3397059  0.36026651  2.4364815 -0.55898773 -0.40568466 -0.57786607
          not        now         of         on         one        only
1 -0.04176153  0.1049530  0.3506382 -0.5643201 -0.31726578 -0.07651245
2 -0.58955133  0.3780618 -0.2435579  0.2131969  0.09429220 -0.62459918
3  0.29793854 -0.2331719 -0.2382584  1.0056141  0.07239926  0.38610287
4  0.31389252 -0.8475975 -1.9330044 -0.1302955  2.95485737  0.28092370
           or         our      shall     should          so       some
1  0.05350945  0.01390439  0.1675906  0.2178890 -0.05937977 -0.2618477
2 -0.38365788 -0.26303984 -0.3792315 -0.2724530 -0.08521280  0.6392920
3 -0.19677510 -0.08883363 -0.2231519 -0.2394768  0.06804264  0.2661959
4  1.62035105  1.09646083  0.4545054 -0.3196166  0.50894675 -0.3439797
         such        than          that        the        their
1  0.07150055 -0.02059864  0.2084314394  0.1159819 -0.273306977
2 -0.52709523 -0.50106422 -0.5238465684 -0.4025561  0.991849610
3 -0.02595886 -0.09047437 -0.1592967827  0.3611036  0.008301811
4  0.78300868  2.15765712 -0.0006027944 -2.3933839  0.365632943
         then      there      things       this         to          up
1 -0.07793873  0.4669874  0.20187475  0.3149081  0.6179541  0.16561596
2 -0.04346848 -0.6560934 -0.42002725 -0.3731165 -0.9652637  0.03601269
3  0.12357132 -0.6633081 -0.09329193 -0.1353435 -0.7179521 -0.34049549
4  0.27440577  0.4137506 -0.60673314 -1.7833116 -0.1442837  0.03935447
        upon         was        were       what        when       which
1  0.6879288 -0.23977418 -0.22843317  0.1146205  0.05561194  0.03705055
2 -0.3728410  1.38567795  1.19163527 -0.3776256 -0.24797150 -0.15054096
3 -1.0156057 -0.08612244 -0.11254063 -0.1307965 -0.13136922  0.15286819
4 -0.7922336 -0.53647667  0.02522827  0.5051132  0.83059962 -0.92930284
          who        will        with      would        your
1  0.03114336 -0.05828478 -0.11207025  0.2473179  0.02897163
2  0.27465588 -0.84541229  0.17875034 -0.3482766 -0.20876463
3 -0.16193522  0.37885591 -0.09147142 -0.5037495 -0.06260916
4 -0.14184107  0.72202330  1.34607296  1.1361017  0.61658391
```

Figure 7

Parin Patel

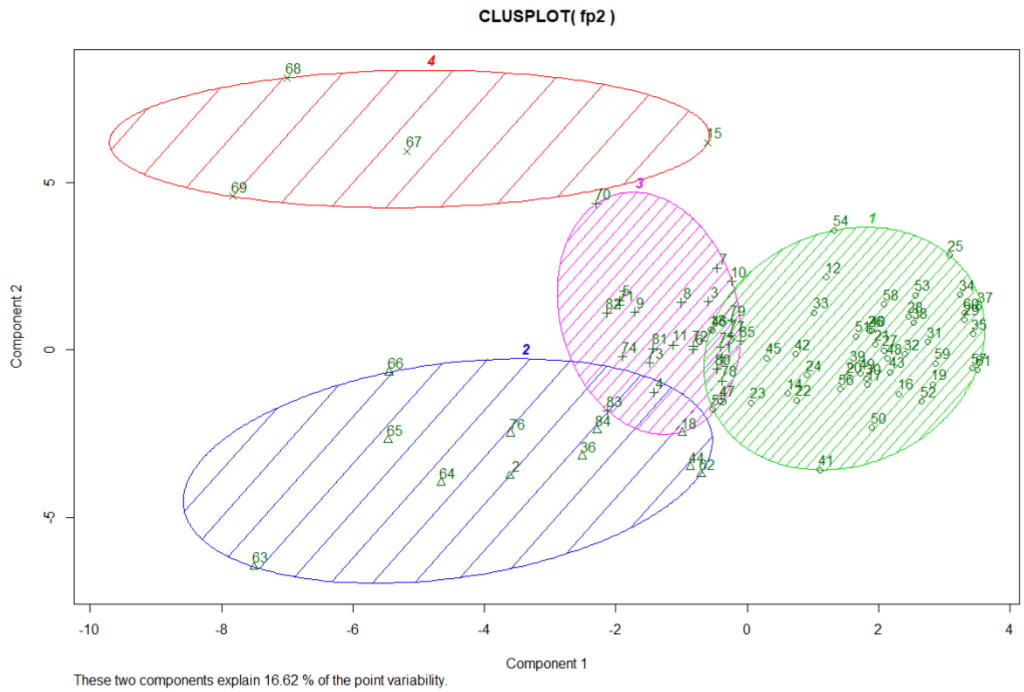


Figure 8

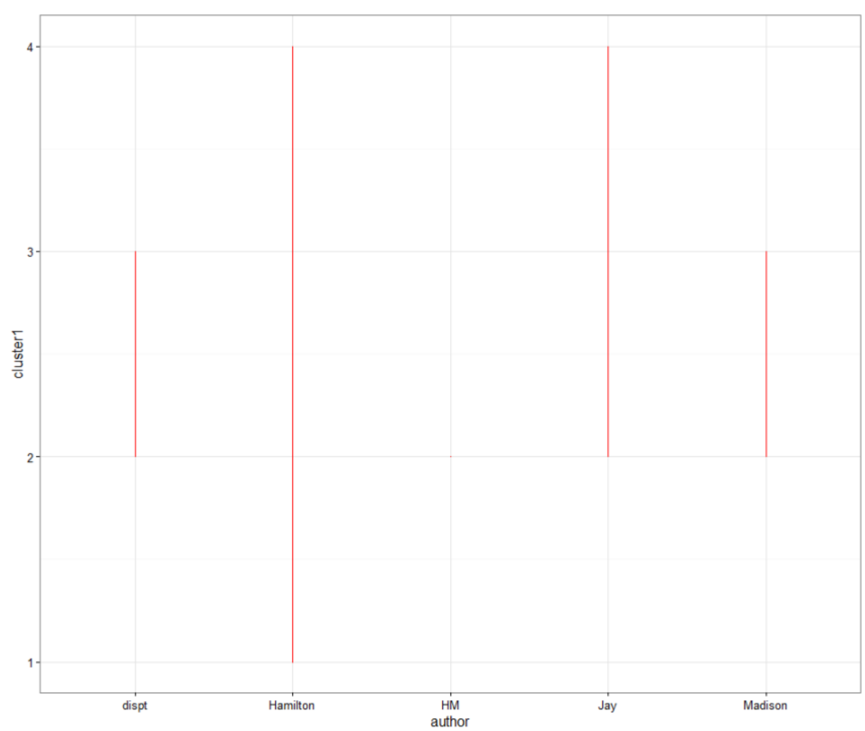


Figure 9

We can use the Figure 9-line plot to see that the disputed authors are divided into clusters 2 and 3. So when compared to the lines for Hamilton and Madison, it is clear that the disputed author line is most similar to Madison. Therefore, it is more likely that the disputed author papers are from Madison.

HAC Algorithm:

In the second algorithm, HAC, we initially tried two different methods. First, we set the method to average and got the results shown in figure 10 and 11. In these outputs, we saw that most of the data was divided into one cluster. This is why we chose to use the complete agglomeration method.

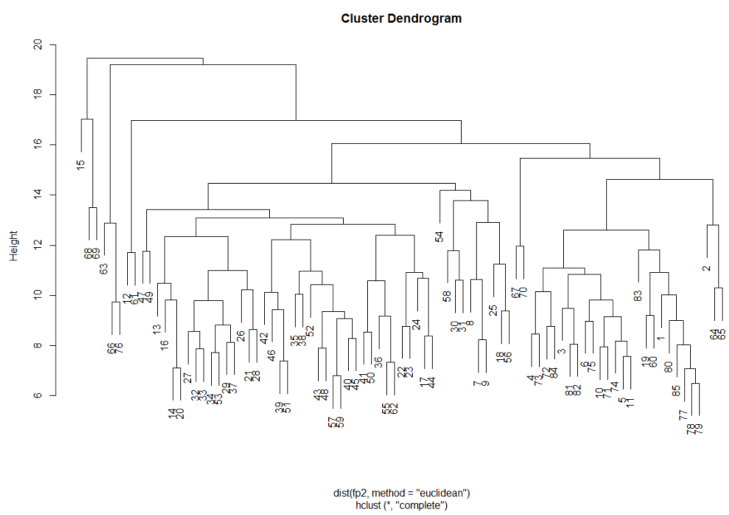


Figure 10

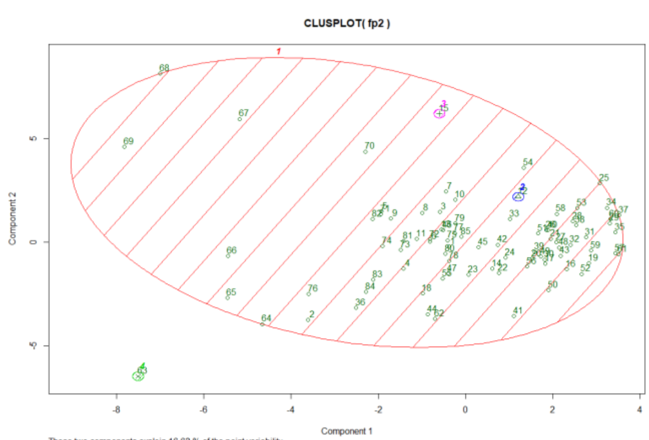


Figure 11

In figure 12 and 13, when we set the agglomeration method to complete, we could see that almost all the disputed papers (1-11) are clustered [located] with Madison's papers (which are 71-85). This confirms the results of the K_means clustering.

Parin Patel

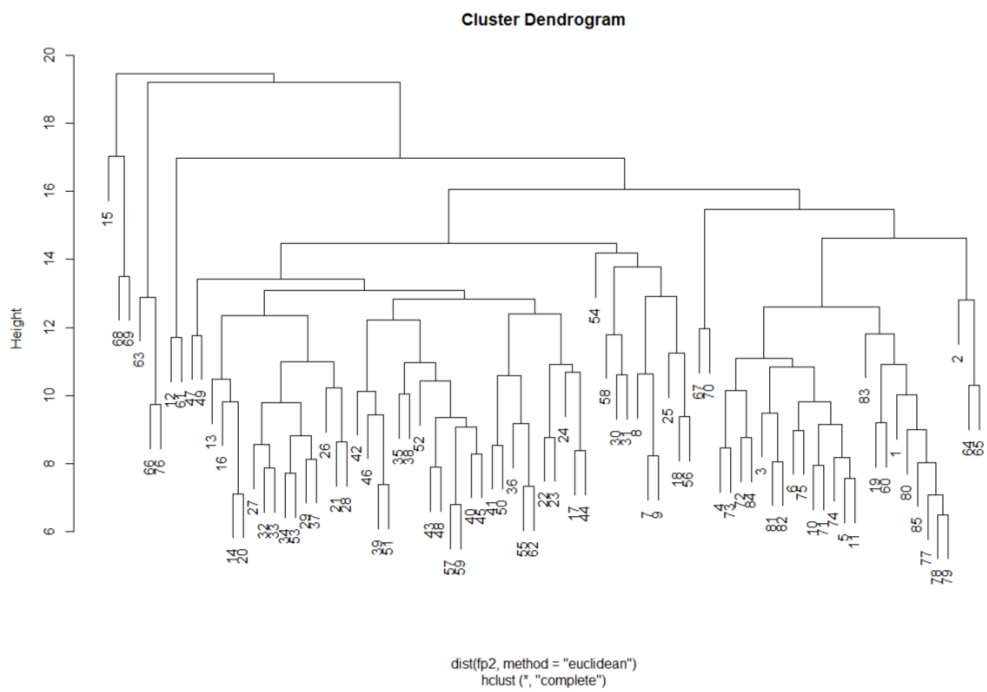


Figure 12

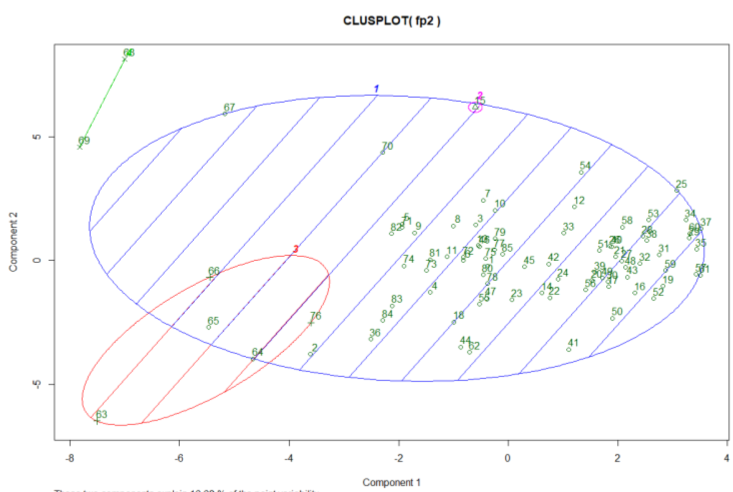


Figure 13

Expectation Maximum (EM) Algorithm:

We used the following resources to understand EM: http://rstudio-pubs-static.s3.amazonaws.com/154174_78c021bc71ab42f8add0b2966938a3b8.html and https://rdrr.io/cran/mclust/man/mclustBIC.html

For the third algorithm, we first installed the mclust library in order to first visualize the clusters (Figure 14). We used the Gaussian finite mixture model, which was fitted by the EM algorithm,

```
####3rd Algorithm: EM#####
install.packages("mclust")
library(mclust)

#visualize clusters
clPairs(fp2,data$author)
fit<-Mclust(fp2)
summary(fit)
```

Figure 14

Since we first wanted to visualize the datapoints. We used the clPairs to see the spectrum for the first 30 columns. We mainly could not increase the number of columns due to the size restraints (Figure 15).

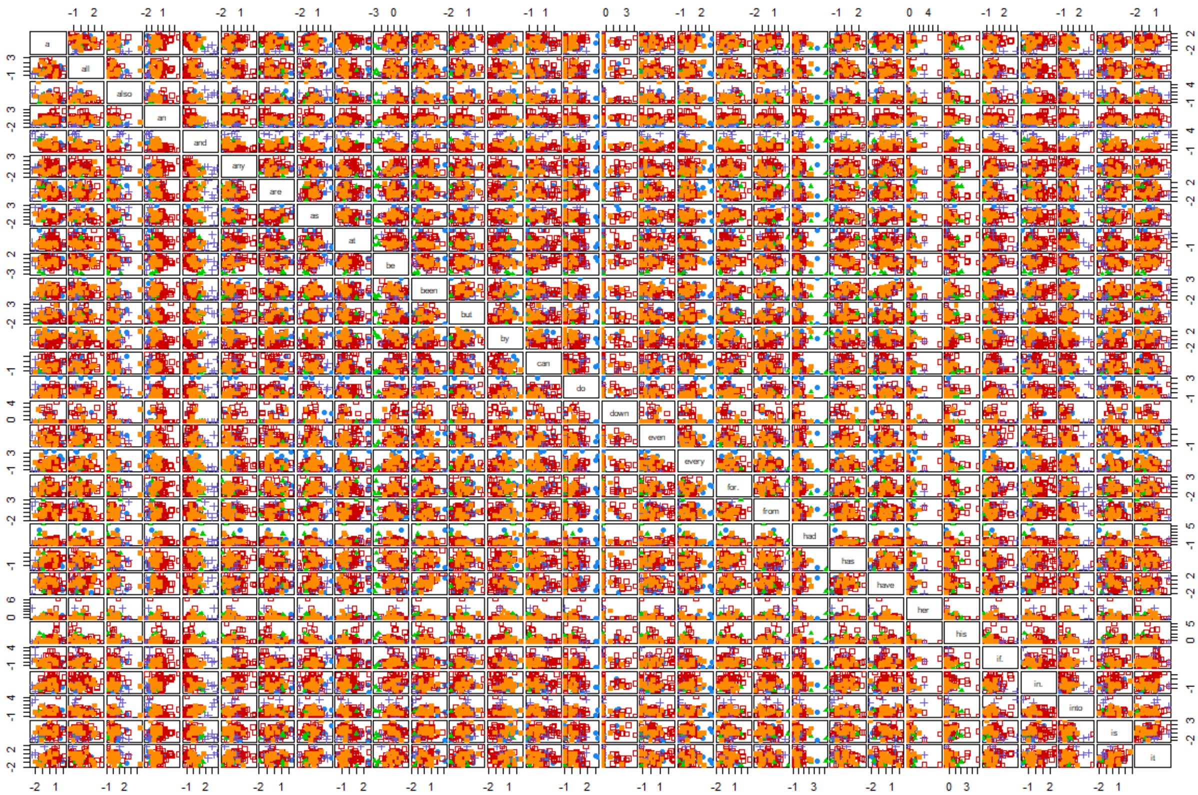


Figure 15

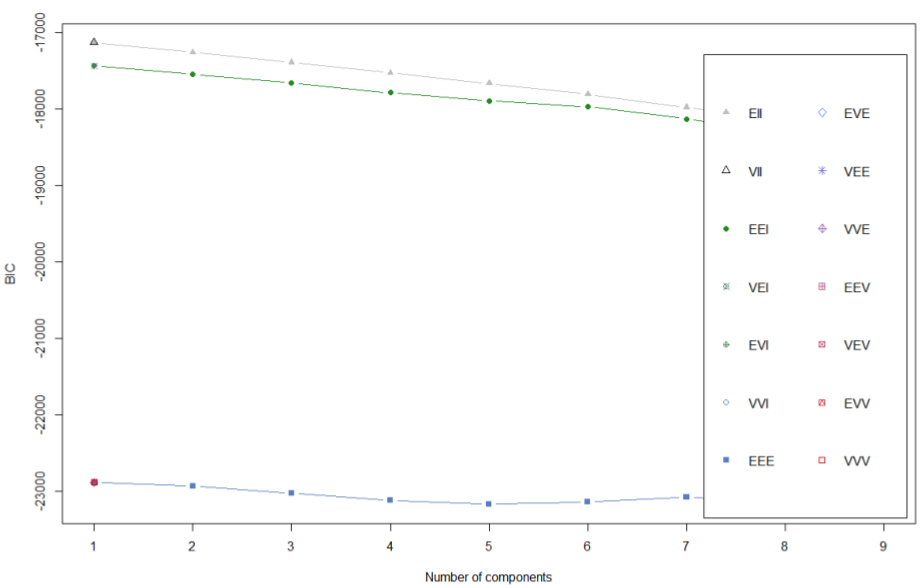
Next, we calculated the fit of the model with one component. The log.likelihood is -8407.477. This is the log likeligood of the BIC Value, which is used for choosing clusters. However, when we plotted our data, the EM algorithm divided all the data points into one cluster, this made it difficult to tell the disputed authors.

```
> fit<-Mclust(fp2)
> summary(fit)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust XII (spherical multivariate normal) model with 1 component:

 log.likelihood  n df       BIC       ICL
      -8407.477 85 71 -17130.38 -17130.38

Clustering table:
 1
85
```



Conclusion:

While the Federalist Papers remain, an important document related to the history of founding and ratification of the US Constitution, understanding who the author of the 11 disputed papers is still considered somewhat of a mystery. While our results were able to, for the most part, show a stronger likelihood that the author was Madison, the results are still not conclusive. One of the main questions raised during this analysis was if the data could be displayed in a hierarchical manner, where records are connected through links. Instead, maybe modeling the papers in a network-like way might be better. Additionally, it is also possible that the papers were written together, by Madison and Hamilton, or even with the assistance of John Jay.

In conclusion, while we did find that the disputed papers are likely the product of Madison, it is also possible that he had help. Further modeling and analysis is required.

Parin Patel

Appendix A: Results

Option 1 Results :

```
> #### Option 1: K means_Clustering ####
> k_mean<-kmeans(fp2, centers = 4, nstart = 25)
> k_mean
K-means clustering with 4 clusters of sizes 24, 4, 11, 46

Cluster means:
            a          all        also         an        and         any
1 -0.06274336  0.038704552  0.41717596 -0.5024666  0.1495853 -0.21337796
2 -1.37026902 -0.617157145  1.58662115 -1.3968214  2.0034108  0.06092825
3 -0.57598787  0.176223940 -0.07835166 -0.3045733  0.7401337 -0.88866997
4  0.28962572 -0.008668348 -0.33688781  0.4564520 -0.4292426  0.31853756
         are         as         at         be        been         but
1  0.4019137  0.3655169 -0.3934412  0.2648322 -0.05293925  0.29481729
2 -0.2705511  0.1120563 -0.9736815  0.1339390 -1.28794190  0.53231438
3 -0.2474613 -0.3541919  0.5272837 -1.3393966  0.27336121 -0.64897176
4 -0.1269924 -0.1157504  0.1638521  0.1704703  0.07424644 -0.04491703
          by         can          do       down        even      every
1  0.6611626  0.07030163 -0.02037322 -0.0890032 -0.07851415  0.6725072
2 -0.1095873  0.27450777  0.68373426 -0.4239542 -0.36977631 -0.6894921
3  0.4744678 -0.88893879 -0.67604087 -0.4239542 -0.39970839 -0.6189158
4 -0.4488847  0.15202297  0.11283630  0.1846824  0.16870081 -0.1429159
         for.         from        had         has       have         her
1  0.06123478 -0.286643438 -0.2738688 -0.16743504 -0.1427725 -0.00244272
2 -0.44859010  0.869797219 -0.6403774 -0.88956500 -1.0360062  2.23104823
3 -0.06056271  0.008195461  1.6808436  0.01947992  0.4388495  0.02101943
4  0.02154164  0.071958773 -0.2033678  0.16005265  0.0596352 -0.19775612
         his         if.        in.         into         is          it
1 -0.2136100 -0.27657201 -0.4937745 -0.009233076  0.2396741 -0.1122614
2 -0.4497950  1.75321470 -1.0246517  2.245473953 -1.1370926  0.2901717
3  0.2055717 -0.41580531 -0.5515928  0.392435886 -0.8577637 -0.7161222
4  0.1014028  0.09127669  0.4786243 -0.284284711  0.1789476  0.2045855
         its         may       more        must          my          no
1 -0.2473462  0.31909596  0.1948770  0.25975274 -0.22413956  0.32764604
2 -0.3397059  0.36026651  2.4364815 -0.55898773 -0.40568466 -0.57786607
3 -0.3206233 -1.11663332 -0.1067859 -0.77537462 -0.02090334 -0.77992250
4  0.2352606  0.06920864 -0.2880071  0.09850013  0.15721793  0.06580667
          not        now        of         on        one        only
1  0.29793854 -0.2331719 -0.2382584  1.0056141  0.07239926  0.38610287
2  0.31389252 -0.8475975 -1.9330044 -0.1302955  2.95485737  0.28092370
3 -0.58955133  0.3780618 -0.2435579  0.2131969  0.09429220 -0.62459918
4 -0.04176153  0.1049530  0.3506382 -0.5643201 -0.31726578 -0.07651245
           or         our      shall     should          so        some
1 -0.19677510 -0.08883363 -0.2231519 -0.2394768  0.06804264  0.2661959
2  1.62035105  1.09646083  0.4545054 -0.3196166  0.50894675 -0.3439797
3 -0.38365788 -0.26303984 -0.3792315 -0.2724530 -0.08521280  0.6392920
4  0.05350945  0.01390439  0.1675906  0.2178890 -0.05937977 -0.2618477
         such        than          that        the        their
1 -0.02595886 -0.09047437 -0.1592967827  0.3611036  0.008301811
2  0.78300868  2.15765712 -0.0006027944 -2.3933839  0.365632943
3 -0.52709523 -0.50106422 -0.5238465684 -0.4025561  0.991849610
4  0.07150055 -0.02059864  0.2084314394  0.1159819 -0.273306977
         then      there      things       this         to          up
1  0.12357132 -0.6633081 -0.09329193 -0.1353435 -0.7179521 -0.34049549
2  0.27440577  0.4137506 -0.60673314 -1.7833116 -0.1442837  0.03935447
3 -0.04346848 -0.6560934 -0.42002725 -0.3731165 -0.9652637  0.03601269
4 -0.07793873  0.4669874  0.20187475  0.3149081  0.6179541  0.16561596
        upon         was        were       what        when       which
1 -1.0156057 -0.08612244 -0.11254063 -0.1307965 -0.13136922  0.15286819
2 -0.7922336 -0.53647667  0.02522827  0.5051132  0.83059962 -0.92930284
3 -0.3728410  1.38567795  1.19163527 -0.3776256 -0.24797150 -0.15054096
4  0.6879288 -0.23977418 -0.22843317  0.1146205  0.05561194  0.03705055
          who        will        with      would        your
```

Option 2 Results :

```
> km_output1
K-means clustering with 4 clusters of sizes 46, 11, 24, 4

Cluster means:
            a          all        also         an        and         any
1  0.28962572 -0.008668348 -0.33688781  0.4564520 -0.4292426  0.31853756
2 -0.57598787  0.176223940 -0.07835166 -0.3045733  0.7401337 -0.88866997
3 -0.06274336  0.038704552  0.41717596 -0.5024666  0.1495853 -0.21337796
4 -1.37026902 -0.617157145  1.58662115 -1.3968214  2.0034108  0.06092825
         are         as         at         be        been         but
1 -0.1269924 -0.1157504  0.1638521  0.1704703  0.07424644 -0.04491703
2 -0.2474613 -0.3541919  0.5272837 -1.3393966  0.27336121 -0.64897176
3  0.4019137  0.3655169 -0.3934412  0.2648322 -0.05293925  0.29481729
4 -0.2705511  0.1120563 -0.9736815  0.1339390 -1.28794190  0.53231438
          by         can          do       down        even      every
1 -0.4488847  0.15202297  0.11283630  0.1846824  0.16870081 -0.1429159
2  0.4744678 -0.88893879 -0.67604087 -0.4239542 -0.39970839 -0.6189158
3  0.6611626  0.07030163 -0.02037322 -0.0890032 -0.07851415  0.6725072
4 -0.1095873  0.27450777  0.68373426 -0.4239542 -0.36977631 -0.6894921
         for.         from        had         has       have         her
1  0.02154164  0.071958773 -0.2033678  0.16005265  0.0596352 -0.19775612
2 -0.06056271  0.008195461  1.6808436  0.01947992  0.4388495  0.02101943
3  0.06123478 -0.286643438 -0.2738688 -0.16743504 -0.1427725 -0.00244272
4 -0.44859010  0.869797219 -0.6403774 -0.88956500 -1.0360062  2.23104823
         his         if.        in.         into         is         it
1  0.1014028  0.09127669  0.4786243 -0.284284711  0.1789476  0.2045855
2  0.2055717 -0.41580531 -0.5515928  0.392435886 -0.8577637 -0.7161222
3 -0.2136100 -0.27657201 -0.4937745 -0.009233076  0.2396741 -0.1122614
4 -0.4497950  1.75321470 -1.0246517  2.245473953 -1.1370926  0.2901717
         its         may       more        must          my          no
1  0.2352606  0.06920864 -0.2880071  0.09850013  0.15721793  0.06580667
2 -0.3206233 -1.11663332 -0.1067859 -0.77537462 -0.02090334 -0.77992250
3 -0.2473462  0.31909596  0.1948770  0.25975274 -0.22413956  0.32764604
4 -0.3397059  0.36026651  2.4364815 -0.55898773 -0.40568466 -0.57786607
          not        now         of         on         one        only
1 -0.04176153  0.1049530  0.3506382 -0.5643201 -0.31726578 -0.07651245
2 -0.58955133  0.3780618 -0.2435579  0.2131969  0.09429220 -0.62459918
3  0.29793854 -0.2331719 -0.2382584  1.0056141  0.07239926  0.38610287
4  0.31389252 -0.8475975 -1.9330044 -0.1302955  2.95485737  0.28092370
           or         our      shall     should          so       some
1  0.05350945  0.01390439  0.1675906  0.2178890 -0.05937977 -0.2618477
2 -0.38365788 -0.26303984 -0.3792315 -0.2724530 -0.08521280  0.6392920
3 -0.19677510 -0.08883363 -0.2231519 -0.2394768  0.06804264  0.2661959
4  1.62035105  1.09646083  0.4545054 -0.3196166  0.50894675 -0.3439797
         such        than          that        the        their
1  0.07150055 -0.02059864  0.2084314394  0.1159819 -0.273306977
2 -0.52709523 -0.50106422 -0.5238465684 -0.4025561  0.991849610
3 -0.02595886 -0.09047437 -0.1592967827  0.3611036  0.008301811
4  0.78300868  2.15765712 -0.0006027944 -2.3933839  0.365632943
         then      there      things       this         to          up
1 -0.07793873  0.4669874  0.20187475  0.3149081  0.6179541  0.16561596
2 -0.04346848 -0.6560934 -0.42002725 -0.3731165 -0.9652637  0.03601269
3  0.12357132 -0.6633081 -0.09329193 -0.1353435 -0.7179521 -0.34049549
4  0.27440577  0.4137506 -0.60673314 -1.7833116 -0.1442837  0.03935447
        upon         was        were       what        when       which
1  0.6879288 -0.23977418 -0.22843317  0.1146205  0.05561194  0.03705055
2 -0.3728410  1.38567795  1.19163527 -0.3776256 -0.24797150 -0.15054096
3 -1.0156057 -0.08612244 -0.11254063 -0.1307965 -0.13136922  0.15286819
4 -0.7922336 -0.53647667  0.02522827  0.5051132  0.83059962 -0.92930284
          who        will        with      would        your
1  0.03114336 -0.05828478 -0.11207025  0.2473179  0.02897163
2  0.27465588 -0.84541229  0.17875034 -0.3482766 -0.20876463
3 -0.16193522  0.37885591 -0.09147142 -0.5037495 -0.06260916
4 -0.14184107  0.72202330  1.34607296  1.1361017  0.61658391
```

Appendix B: R Code

```
data_path=file.choose()
data=read.csv(data_path)
str(data)
sum(!complete.cases(data))
summary(data)
fp<-data[,c(-1,-2)]

#Scaling fp data
```

Parin Patel

```
fp2<-data.frame(scale(fp,center = T,scale = T))





#Plot (elbow method) to decide optimal number of clusters
set.seed(10)
optim.cluster<-function(k){
  return(kmeans(fp2,k,nstart = 25)$tot.withinss)
}
k_values<-1:14
oc_values<-purrr::map_dbl(k_values,optim.cluster)
plot(x=k_values, y=oc_values,type="b",frame=F,xlab = "Number of clusters K",ylab="Total within-clusters (sum squared)")


#### Option 1: K means_Clustering ####
k_mean<-kmeans(fp2, centers = 4, nstart = 25)
k_mean
k_mean$centers


######Option 2: :K_means using Hartingan-Wong Method####

##HW algorithm to find centroid
km_output1<-kmeans(fp2,centers=4,nstart = 25,iter.max = 100,algorithm = "Hartigan-Wong")
km_output1
km_output1$centers



#visualize the result
install.packages("cluster")
library(cluster)
install.packages("ggplot2")
library(ggplot2)

clusplot(fp2,km_output1$cluster,color=TRUE,shade=T,labels = 2,lines =0 )

km_df1<-data.frame(author=data$author,cluster1=km_output1$cluster)
ggplot(km_df1)+geom_polygon(aes(x=author,y=cluster1,group=author,fill=as.factor(cluster1)),color="red")+
  coord_fixed(1.3)+
  guides(fill=F)+
```

Parin Patel

```
  theme_bw()

##### 2nd Algorithm: HAC######
hac_output<-hclust(dist(fp2,method = "euclidean"),method = "average") #average
hac_output2<-hclust(dist(fp2,method = "euclidean"),method = "complete") #complete


#plot HAC
plot(hac_output) ##average
plot(hac_output2) ## complete


#output desirable number of clusters after modeling

#average linkage
hac_cut<-cutree(hac_output,4)
hac_df1<-data.frame(author=data$author,cluster=hac_cut)
avg_Clus_plot<-clusplot(fp2,hac_cut,color=TRUE,shade=T,labels = 2,lines =0 )

#compelte linkage
hac_cut<-cutree(hac_output2,4)
hac_df1<-data.frame(author=data$author,cluster=hac_cut)
clusplot(fp2,hac_cut,color=TRUE,shade=T,labels = 2,lines =0 )




####3rd Algorithm: EM#####
install.packages("mclust")
library(mclust)

#visualize clusters
clPairs(fp2 [1:30],data$author) ##disputed


fit<-Mclust(fp2)
summary(fit)

#1. BIC (The Bayesian information criterion (BIC)
## is used my mclust with is a test used to assess the fit of a model)
plot(fit,what= "BIC")
#2. classification
plot(fit, what = "classification")
length(fit$classification)
```