Clustering & Text, Part 2

Josh Clinton, Vanderbilt University

2022-07-12

Text As Data

One prominent example of unsupervised learning is in terms of the analysis of text-as-data. We see this when using data science methods to describe and understand the "sentiments" being expressed on Twitter, or the content and bias in reporting, and even questions related to prediction: e.g., who wrote a piece of text (and was it plagarized)?

In what follows we are going to expand our focus on unsupervised learning methods – `kmeans` applying it to text data to give you a sense of how the basic algorithm has applicability beyond what you might typically think of as data. It also helps illustrate how much of what we do as data scientists depends on the choices that we make in terms of measurement.

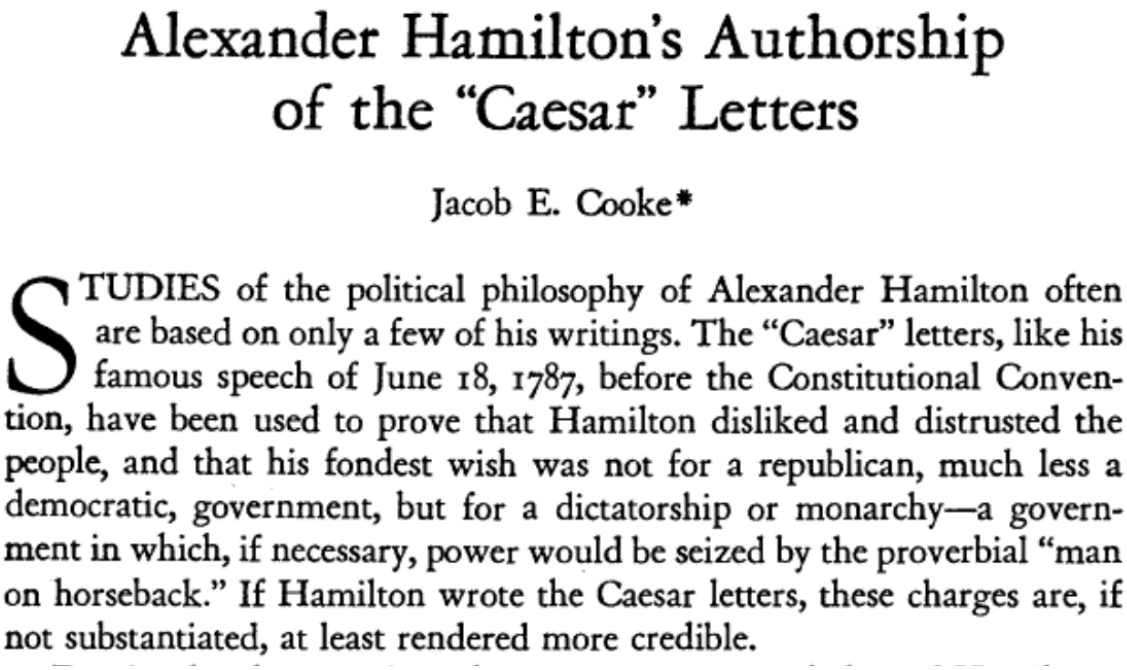
The text we are going to focus on is the text of what are called "The Federalist" papers – a set of 85 essays promoting the ratification of the United States Constitution to the people of NY State written by Alexander Hamilton, James Madison, and John Jay between 1787-1788.

The underlying story was partially recounted in the much more interesting depiction on Broadway

There are two basic questions that we can use data science methods to engage.

First, what are the federalist papers about? What did the authors think that they needed to say to convince the delegates to the New York State Constitutional Convention? How much of what issues and concerns? This is an application of unsupervised learning because we are going to use the algorithm to learn about the patterns in the data rather than to tell the algorithm which patterns to predict.

Second, there is a debate about the authorship of 11 Federalist papers. The author of every Federalist Paper wrote under a pseudo-name to protect their identity – and also try to protect their identity in case they changed their mind. Perhaps Caesar was not the best choice for someone writing about the benefits of democracy? Just saying. . . .



Alexander Hamilton's Authorship of the "Caesar" Letters

Jacob E. Cooke*

STUDIES of the political philosophy of Alexander Hamilton often are based on only a few of his writings. The "Caesar" letters, like his famous speech of June 18, 1787, before the Constitutional Convention, have been used to prove that Hamilton disliked and distrusted the people, and that his fondest wish was not for a republican, much less a democratic, government, but for a dictatorship or monarchy—a government in which, if necessary, power would be seized by the proverbial "man on horseback." If Hamilton wrote the Caesar letters, these charges are, if not substantiated, at least rendered more credible.

But Hamilton's duel and his desire to claim ownership over his legacy meant that he "leaked" a copy of the papers with a list of who he claimed had written each paper. Madison disagreed with Hamilton's claim on 8 of the 85 papers and it was impossible for the two to reconcile due to Hamilton's death. But can we use the authorship of known and uncontested Federalist Papers to predict the authorship of contested papers? This pivots from unsupervised learning to the issue of supervised learning. How can we use known information to "supervise" the creation of a prediction model and make a prediction about an unknown variable. To do so we are going to use a very basic supervised learning algorithm - the liner regression.

Learning about text using data

Text data is called a "corpus". Let's load the complete set of Federalist Papers that are saved as a tidy-format corpus.

```
library("SnowballC")
library("tidyverse")
```

```
## Warning: replacing previous import 'lifecycle::last_warnings' by
## 'rlang::last_warnings' when loading 'pillar'
```

```
## Warning: replacing previous import 'lifecycle::last_warnings' by
## 'rlang::last_warnings' when loading 'hms'
```

```
## Warning: package 'tibble' was built under R version 4.1.2
```

```
library("tidytext")
library("tm")

corpus.tidy <- readRDS(file="FederalistPaperCorpusTidy.Rds")
glimpse(corpus.tidy)
```

```
## Rows: 85
## Columns: 3
```

```
## $ author   <chr> "hamilton", "jay", "jay", "jay", "jay", "hamilton", "hamilton~
## $ text     <chr> "after an unequivocal experience of the inefficiency of the s~
## $ document <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18~
```

So there are 85 observations – one for each Federalist paper (indexed by `document`) and each document has an associated `author` as well as a `text` containing the entire text of the document. So if we look at the first observation of the `text` variable we can see the entirity of Federalist Paper 1 (with line breaks denotes by `\n`). Note that we have preprocessed the document to remove all numbers, punctuations, and capitalizations so that the entry contains only the words contained in each.

```
corpus.tidy$text[1]
```

To analyze this we need to convert the text into `tokens` that we can analyze by breaking apart the characters in each `text` into the separate words. To do so we are going to do the following manipulation:

```
tokens <- corpus.tidy %>%
  # tokenizes into words and stems them
  unnest_tokens(word, text, token = "word_stems") %>%
  # remove any numbers in the strings
  mutate(word = str_replace_all(word, "\\d+", "")) %>%
  # drop any empty strings
  filter(word != "")
```

So the `unnest_tokens` breaks the document up into words and extracts the "stem" - i.e, the portion of the work with lexical meaning (e.g., not suffixes or prefixes) – which we then `mutate` to replace all numbers in the string with an empty string and then we conclude by filtering out the empty strings.

So what did this give us?

```
tokens
```

```
## # A tibble: 187,105 x 3
##    author   document word
##    <chr>       <int> <chr>
##  1 hamilton        1 after
##  2 hamilton        1 an
##  3 hamilton        1 unequivoc
##  4 hamilton        1 experi
##  5 hamilton        1 of
##  6 hamilton        1 the
##  7 hamilton        1 ineffici
##  8 hamilton        1 of
##  9 hamilton        1 the
## 10 hamilton        1 subsist
## # ... with 187,095 more rows
```

So this `tidy` tibble has every observation in a word in a document. There are therefore 187,105 words that are used in the 85 federalist papers.

But what are the Federalist Papers about? Can we deduce the meaning based on the words being used? To start, what if we look at the distributions of words to characterize meaning. To do so let's create a table of the frequency of each word sorted in decreasing order (i.e., `arrange(-n)`):

```
tokens %>%
  count(word) %>%
  arrange(-n)
```

```
## # A tibble: 4,980 x 2
##    word      n
```

```
##    <chr> <int>
##  1 the   17388
##  2 of    11507
##  3 to     6905
##  4 and    5032
##  5 in     4390
##  6 a      3954
##  7 be     3927
##  8 it     3163
##  9 that   2750
## 10 is     2160
## # ... with 4,970 more rows
```

Hmm. So the Federalist Papers are about "the" and "of" and "to"? It makes sense that the transition words are most-frequently used, but let's strip out these words that are not useful for understanding content. To do so we are going to read in the dataframe `stop_words` that is contained in the package `tidytext`. This is a predefined dictonary of commonly used ords in the English language that provides a standard set of words that can be used to prune the text and remove them from the analysis.

Note that the `anti_join` is essentially going thru the tibble `tokens` to remove (i.e., "anti_join") any observations contained in the tibble `stop_words`. (The `by` defines how we are going to try to match the tibble `tokens` to `stop_words` – here we are going to look word-by-word for matches to remove.)

```
data("stop_words", package = "tidytext")
tokens <- anti_join(tokens, stop_words, by = "word")
```

So now let us see what is left and how that compares to the table we created before removing the "stop words" - i.e., all of the little words we use in English to combine ideas (e.g., the, and, or, all). Basically words that are used in nearly every expression of written English regardless of topic or content.

```
tokens %>%
  count(word) %>%
  arrange(-n)
```

```
## # A tibble: 4,675 x 2
##    word          n
##    <chr>     <int>
##  1 govern     1026
##  2 power       905
##  3 constitut   672
##  4 nation      566
##  5 ani         540
##  6 peopl       522
##  7 author      393
##  8 object      375
##  9 union       361
## 10 everi       348
## # ... with 4,665 more rows
```

Much better, and much different!

Now lets use this tibble to create a new tibble that is a count by document and word stem. The tibble resulting `dtm` – document-term matrix, not "dead-to-me" – has each observation as a word-stem in a document with the variable `n` denoting the frequency with which each word stem appears.

```
dtm <- tokens %>%
  count(document, word)
```

```
glimpse(dtm)
```

```
## Rows: 38,829
## Columns: 3
## $ document <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ word     <chr> "abl", "absurd", "accid", "accord", "acknowledg", "act", "act~
## $ n        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 2, 1, 1, 1, 2, 1~
```

So using this we can then try to visualize the most-frequently used words using a `wordcloud`. To produce a wordcloud we need to provide a list of potential words and the frequency which which each word appears (`freq`) and then we can also limit the visualization to include words that are mentioned at least `min.freq` times such that no more than `max.words` are plotted in the resulting word cloud.

So if we did not like the table we produced earlier of the most frequently used words we could also depict this graphically. So what are the federalist papers about?

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

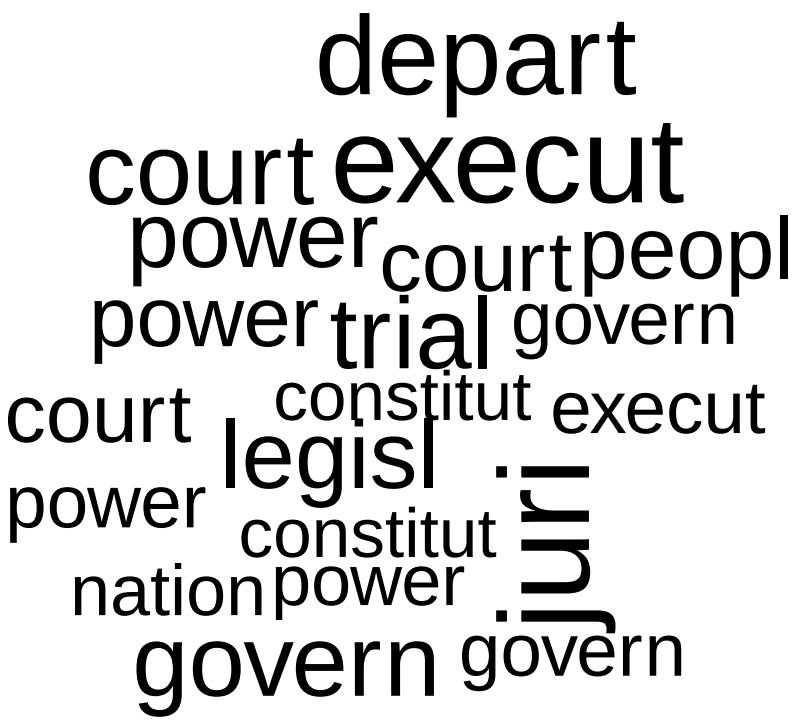
```
wordcloud(words = dtm$word,
          freq = dtm$n,
          min.freq = 20,
          max.words = 200,
          random.order=FALSE,
          rot.per=0.35)
```

NOTE: How does your understanding/impression/characterization change as you change these parameters? Try it out. What does that mean in terms of the usefulness of a wordcloud?

```
# INSERT CODE HERE
```

As an aside, if you wanted to use piping, you could do the same using the following code – here using only the top 20 words. Note that the `{}` is required to deal with the fact that we are passing `wordcloud` thru `dtm` and making an internal reference being reflected in the use of `.`

```
dtm %>%
    { wordcloud(.$word, .$n, max.words = 20) }
```



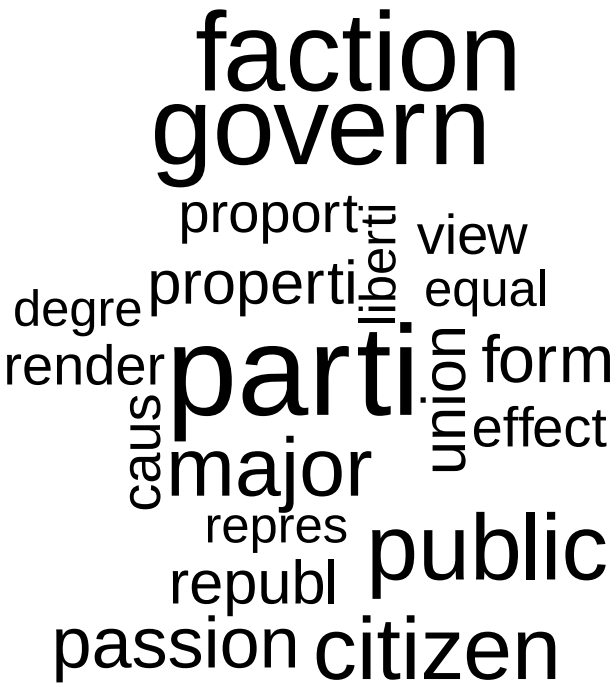
So what if we wanted to focus on a particular paper – say Federalist 10? One way is to create a new tibble containing just Federalist 10. This may be useful if you were going to do a lot of subsequent work focusing on Federalist 10 and you wanted to avoid having to filter every time.

```
dtm10 <- dtm %>%
  filter(document == 10)  %>%
  arrange(-n)

wordcloud(words = dtm10$word,
          freq = dtm10$n,
          min.freq = 3,
          max.words = 50,
          random.order=FALSE,
          rot.per=0.35)
```

But we could also produce a `wordcloud` of the top 20 words in Federalist 10 by applying it to the appropriately `filtered` `dtm` tibble:

```
dtm %>%
  filter(document == 10) %>% {
    wordcloud(.$word, .$n, max.words = 20)
  }
```



So what are the 10 most frequently used words?

```
dtm10 %>%
  top_n(10,wt=n)
```

```
## # A tibble: 11 x 3
##    document word         n
##       <int> <chr>    <int>
##  1       10 parti       20
##  2       10 faction     17
##  3       10 govern      17
##  4       10 public      14
##  5       10 citizen     13
##  6       10 major       12
##  7       10 passion     10
##  8       10 form         9
##  9       10 properti     8
## 10       10 republ       8
## 11       10 union        8
```

Summarizing/Classifying Content

How else can we summarize/describe data? Cluster Analysis via `kmeans`?

But using what data? Should we focus on the number of words being used? The proportion of times a word is used in a particular document? Or some other transformation that tries to account for how frequently a word is used in a particular document relative to how frequently it is used in the overall corpus?

We are going to use the text analysis function `bind_tf_idf` that will take a document-term matrix and compute the fraction of times each word is used in each document (`tf` = "term frequency"). It also computes a transformation called tf-idf that balances how frequently a word is used relative to its uniqueness in a document.

For word `w` in document `d` we can compute the tf-idf using:

$$tf - idf(w,d) = tf(w,d) \times log\left(\frac{N}{df(w)}\right)$$

where `tf` is the term frequency (word count/total words), `df(w)` is the number of documents in the corpus that contain the word, and `N` is the number of documents in the corpus. The inverse-document-frequency `idf` for each word `w` is therefore the number of documents in the corpus `N` over the number of documents containing the word.

NOTE: In what follows we are going to focus on the `tf` as it is easier to grasp conceptually, but if you are interested you should replicate the analyses using the `tf_idf` transformation to see how measurement choices can matter (i.e., replace `tf` with `tf_idf`).

So let us create an new document-term-matrix object that also includes the `tf`, `idf` and `tf_idf` associated with each word. Using the resulting tibble – `dtm.tfidf` let us look at the values associated with Federalist 10 written by Madison:

```
dtm.tfidf <- bind_tf_idf(dtm, word, document, n) # use special function to bind the information

dtm.tfidf  %>%
  filter(document == 10) %>%
    top_n(10,
          wt=tf_idf)  # Use a weighted sum
```

```
## # A tibble: 10 x 6
##    document word          n      tf   idf  tf_idf
##       <int> <chr>     <int>   <dbl> <dbl>   <dbl>
##  1       10 cure          5 0.00466 2.50  0.0116
##  2       10 democraci     5 0.00466 3.06  0.0142
##  3       10 faction      17 0.0158  1.26  0.0200
##  4       10 factious      4 0.00372 2.65  0.00987
##  5       10 injustic      4 0.00372 2.50  0.00930
##  6       10 major        12 0.0112  1.15  0.0128
##  7       10 parti        20 0.0186  0.705 0.0131
##  8       10 passion      10 0.00931 0.946 0.00881
##  9       10 properti      8 0.00745 1.22  0.00912
## 10       10 republ        8 0.00745 1.18  0.00882
```

So kmeans took a matrix where each column was a different variable that we were interested in using to characterize patterns but the data we have is arranged in a one-term-per-document-per-row. To transform the data into the format we require we need to "recast" the data so that each word is a separate variable – meaning that the number of variables is the number of of unique word stems. H

```
cast_dtm(dtm.tfidf, document, word, tf)
```

```
## <<DocumentTermMatrix (documents: 85, terms: 4675)>>
## Non-/sparse entries: 38829/358546
## Sparsity           : 90%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

So now let us create this recasted object for reach and use it for analysis.

```
castdtm <- cast_dtm(dtm.tfidf, document, word, tf)

set.seed(42)
km_out <- kmeans(castdtm,
                 centers = 4,
                 nstart = 25)
```

So how many documents are associated with each cluster?

```
table(km_out$cluster)
```

```
##
##  1  2  3  4
## 46 29  2  8
```

So let's tidy it up to see the centroids – here mean frequency– associated with each word in each cluster.

```
tidy(km_out)
```

```
## # A tibble: 4 x 4,677
##        abl    absurd    accid  accord acknowledg     act  actuat     add   addit
##      <dbl>     <dbl>    <dbl>   <dbl>      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 0.00158  0.000287   3.84e-5 1.05e-3   0.000480 0.00135 1.64e-4 3.20e-4 9.17e-4
## 2 0.000789 0.0000996  0       6.54e-4   0.000160 0.00243 1.82e-4 7.56e-4 1.06e-3
## 3 0        0          0       1.69e-3   0        0.00458 7.31e-4 0       0
## 4 0        0.000545   1.49e-4 1.31e-3   0.000186 0.00255 0       1.86e-4 5.77e-4
## # ... with 4,668 more variables: address <dbl>, admit <dbl>, adopt <dbl>,
## #   advantag <dbl>, adversari <dbl>, advoc <dbl>, affect <dbl>, afford <dbl>,
## #   aggrand <dbl>, aim <dbl>, alreadi <dbl>, altern <dbl>, alway <dbl>,
## #   ambigu <dbl>, ambit <dbl>, ambiti <dbl>, america <dbl>, amus <dbl>,
```

```
## #   analog <dbl>, angri <dbl>, ani <dbl>, animos <dbl>, anoth <dbl>,
## #   answer <dbl>, antagonist <dbl>, apt <dbl>, ardent <dbl>, argument <dbl>,
## #   arriv <dbl>, artific <dbl>, astray <dbl>, attain <dbl>, attempt <dbl>, ...
```

Very hard to summarize given that there are 4677 columns! (Good luck trying to graph that!)

How can we summarize? Can we `augment`? No because `augment` does not work for DocumentTermMatrix objects. So let's do it by hand.

First we are going to create a new tibble containing all of the unique words in the document-term-matrix object we analyzed called `words_kmean`. Then we are ging to `bind_cols` this to the matrix of centers (i.e., the mean frequency with which each term appears in documents associated with each cluster). So we can see that we have effectively flipped the rows and columns.

```
words_kmean <- tibble(word = colnames(castdtm))
words_kmean <- bind_cols(words_kmean, as_tibble(t(km_out$centers)))  # Binding the transpose of the mat
words_kmean
```

```
## # A tibble: 4,675 x 5
##    word             `1`       `2`       `3`       `4`
##    <chr>          <dbl>     <dbl>    <dbl>    <dbl>
##  1 abl        0.00158   0.000789  0        0
##  2 absurd     0.000287  0.0000996 0        0.000545
##  3 accid      0.0000384 0         0        0.000149
##  4 accord     0.00105   0.000654  0.00169  0.00131
##  5 acknowledg 0.000480  0.000160  0        0.000186
##  6 act        0.00135   0.00243   0.00458  0.00255
##  7 actuat     0.000164  0.000182  0.000731 0
##  8 add        0.000320  0.000756  0        0.000186
##  9 addit      0.000917  0.00106   0        0.000577
## 10 address    0.000369  0.000152  0.00143  0
## # ... with 4,665 more rows
```

So now let us see what the top words are by gathering the words in each cluster according to their value (i.e., average document frequency) and then reporting the top 10 words in each cluster organized in descending overall average document frequency. (TEST: Can you make it report in descending average document frequency by cluster?)

```
top_words_cluster <-
  gather(words_kmean, cluster, value, -word) %>%
  group_by(cluster) %>%
  top_n(10,
        wt=value) %>%
  arrange(-value)

top_words_cluster
```

```
## # A tibble: 40 x 3
## # Groups:   cluster [4]
##    word      cluster  value
##    <chr>     <chr>    <dbl>
##  1 execut    3       0.0438
##  2 depart    3       0.0423
##  3 legisl    3       0.0366
##  4 power     3       0.0330
##  5 constitut 3       0.0268
##  6 power     4       0.0249
```

```
##  7 govern    1       0.0194
##  8 judiciari 3       0.0180
##  9 court     4       0.0175
## 10 law       4       0.0165
## # ... with 30 more rows
```

So let's take a look at each cluster and what the top 10 words might imply about the content of each cluster.

```
top_words_cluster %>%
  filter(cluster==1) %>%
  arrange(-value)
```

```
## # A tibble: 10 x 3
## # Groups:   cluster [1]
##    word      cluster   value
##    <chr>     <chr>     <dbl>
##  1 govern    1       0.0194
##  2 power     1       0.0115
##  3 nation    1       0.0111
##  4 peopl     1       0.00875
##  5 union     1       0.00696
##  6 constitut 1       0.00681
##  7 ani       1       0.00661
##  8 object    1       0.00556
##  9 author    1       0.00544
## 10 feder     1       0.00482
```

Seems to be about power, governance, federalism and the nature of the union proposed by the constitution. Big picture themes and concepts about the role of government and the power of the nation versus states.

```
top_words_cluster %>%
  filter(cluster==2) %>%
  arrange(-value)
```

```
## # A tibble: 10 x 3
## # Groups:   cluster [1]
##    word      cluster   value
##    <chr>     <chr>     <dbl>
##  1 power     2       0.0111
##  2 govern    2       0.0107
##  3 constitut 2       0.0105
##  4 ani       2       0.0103
##  5 repres    2       0.00940
##  6 peopl     2       0.00874
##  7 senat     2       0.00829
##  8 elect     2       0.00718
##  9 execut    2       0.00678
## 10 bodi      2       0.00676
```

Looks like papers about representation and governance. The connections between people and their elected Representatives and (at the time) appointed Senators. Basically the way that the democratic part would work.

```
top_words_cluster %>%
  filter(cluster==3) %>%
  arrange(-value)
```

```
## # A tibble: 10 x 3
```

```
## # Groups:   cluster [1]
##    word      cluster   value
##    <chr>     <chr>     <dbl>
##  1 execut    3        0.0438
##  2 depart    3        0.0423
##  3 legisl    3        0.0366
##  4 power     3        0.0330
##  5 constitut 3        0.0268
##  6 judiciari 3        0.0180
##  7 govern    3        0.0150
##  8 appoint   3        0.0105
##  9 exercis   3        0.00939
## 10 branch    3        0.00888
```

This looks like it is about separation of powers issues. How the executive, legislative and judicial branches were related. Executive appointments and senate confirmation. Within-government relations.

```
top_words_cluster %>%
  filter(cluster==4) %>%
  arrange(-value)
```

```
## # A tibble: 10 x 3
## # Groups:   cluster [1]
##    word      cluster   value
##    <chr>     <chr>     <dbl>
##  1 power     4        0.0249
##  2 court     4        0.0175
##  3 law       4        0.0165
##  4 constitut 4        0.0159
##  5 author    4        0.0144
##  6 nation    4        0.0102
##  7 union     4        0.0100
##  8 jurisdict 4        0.00971
##  9 govern    4        0.00959
## 10 ani       4        0.00950
```

The last set look to be about courts, laws, jurisdictions and power. It looks like they are related to the power and role of the judiciary in the new proposed government.

Can we make this list a bit easier to follow? Of course. Lets summarize and then use `kable` (invoked via `knitr`) to make the resulting list of words more attractive to look at.

```
gather(words_kmean, cluster, value, -word) %>%
  group_by(cluster) %>%
  top_n(10, value) %>%
  arrange(-value) %>%
  summarise(top_words = str_c(word, collapse = ", ")) %>%
  mutate(NumPapers = table(km_out$cluster)) %>%
  knitr::kable()
```

cluster	top_words	NumPapers
1	govern, power, nation, peopl, union, constitut, ani, object, author, feder	46
2	power, govern, constitut, ani, repres, peopl, senat, elect, execut, bodi	29
3	execut, depart, legisl, power, constitut, judiciari, govern, appoint, exercis, branch	2
4	power, court, law, constitut, author, nation, union, jurisdict, govern, ani	8

Really cool. So we have used nothing more than the `kmeans` clustering to uncover the meaning of the Federalist Papers! (Or at least the meaning in terms of the explicit language being used.)

But recall that there were 9 papers were the authorship was unknown because both Hamilton and Madison claimed authorship. Can we use the clusters to identify authorship? Put differently, do the topics we uncover covary with the authors of the papers in ways that would let us use the former to infer the latter? Did the authors separate their writing based on topic content?

Well, let's see how well the clusters correspond to the various authors?

```
table(Cluster = km_out$cluster, Author = corpus.tidy$author)
```

```
##         Author
## Cluster contested hamilton hamilton.madison jay madison
##       1         3       26                3   4      10
##       2         8       19                0   1       1
##       3         0        0                0   0       2
##       4         0        6                0   0       2
```

Not very well. A majority of Hamilton and Madison's known writings were classified in Cluster so it is hard to know how to interpret the 3 contested papers that were similarly classified. Nor does a similar cluster imply authorship – e.g., 4 of the papers written by Jay are in Cluster 1 but it would be wrong to attribute those papers to Hamilton or Madison!

Perhaps the 8 contested paper in Cluster 2 are more suggestive given that the other papers in that cluster were primarily written by Hamilton (19) rather than Jay (1) or Madison (1). However, `kmeans` is intended to summarize within-sample, not predict out-of-sample. So while we might be tempted to make a prediction based on similar content, realize that this is an interpretation that goes beyond what we have used the data to do. We will return to this later.

Before the we can use the same tools and processes to look at what specific authors are writing about. For example, if we wanted to look at Hamilton's topics and Madison's topics we can separately analyze the writings of each. Creating a index to identify the authorship of each doucment we can then filter the `dtm.tdidf` tibble defined about to extract only the papers known to be written by Hamilton (or Madison). We can then replicate what we just did to this subset of Federalist Papers.

```
hamilton <- c(1, 6:9, 11:13, 15:17, 21:36, 59:61, 65:85)
madison <- c(10, 14, 37:48, 58)

dtm_hamilton <- filter(dtm.tfidf, document %in% hamilton)
castHamilton <- cast_dtm(dtm_hamilton, document, word, tf)
```

SELF_TEST: Can you use `castHamilton` to summarize the 4 main topics that Hamilton wrote about? How many papers are in each cluster? What is the primary themes of each?

```
# INSERT CODE
```

SELF_TEST: And how does the emphasis of Hamilton compare to Madison in terms of focus and content?? (Requires creating the equivalent `castMadison` object.)

```
# INSERT CODE
```