Chapter 2: Investigating the Memory of Events Using n-grams, Controlled Vocabulary, and Joins

Computational methods support inquiry into how the discourses and events in the past shape current-day political change. It is common in speeches for political actors to reference the events of the past – whether American politicians reference the Tea Party or British parliamentarians reference the events of the Second World War. Speakers in parliament also acknowledge contemporary public protests as a source of political legitimacy. But the number, intensity, and specific events referenced change over time. What can we learn about how political reforms were made in the past from reviewing how politicians of the past referenced events?

In this chapter, we will explore questions such as these: In the lead-up to the Great Reform Act of 1832, when the middle class got the vote in Britain, which historical references were on the table? Which members of parliament contributed the most to the debate about the Abolition of Slavery in 1833, and of these speakers, whose language reflected the most on the lived experience of toil and cruelty in the system of slavery? Which historical and contemporary events were referenced the most by members of parliament in the era of the Second Reform Act of 1867, when working-class people got the vote for the first time?

This chapter builds upon a sense of how to leverage computational historical thinking to create insight about history. It introduces counting n-grams, or multiword phrases. It also introduces the use of a "controlled vocabulary," that is, a historian-curated list of important words and phrases, to explore which events were spoken about in parliament during the 1830s.

This chapter builds on the previous to introduce analysts to fundamental data processing techniques in R, commonly used for preparing data for analysis. A key task in data analysis is summarizing a dataset's characteristics to gain insights into the features of the corpus. Scholars have drawn attention to how th immediacy of computational text analysis–its perceived automation in transforming a raw, messy corpus into a curated object of analysis–masks the degree of transformation that is enacted on the text's "original" and intermediary data structures (Palermo 2017). In this chapter, we draw attention to each step in data processing, emphasizing how intermediary data wrangling steps contributes to the creation of new knowledge.

One essential concept in data processing is "joining" datasets. A join combines two datasets into one, allowing us to analyze parliamentary texts alongside metadata such as speaker information, dates, and significant events. Additionally, this chapter teaches readers how to work with event-based data, including years, months, and dates. By moving through the data processing steps one-by-one, we draw on Paul Fyfe's framing of "data mining" as "data archaeology," shifting attention away from analysis as a final product and instead foregrounding the transformational work of data processing as a central part of the analytical process (Fyfe 2016).

As with the previous chapter, this chapter will emphasize the skills of moving from analyzing text as data to performing a close reading of historical speeches in context. It models the iterative process of working with a controlled vocabulary and reading words in context with the idea of demonstrating a method of inquiry about how and why events happen. This approach provides analysts with the tools of deciding for themselves how the patterns identified throughout text mining relate back to their primary sources, that

Table 1: Hansard Speeches 1860

sentence_id	text
S3V0156P0_0	which had been prorogued successively to the 27th October, thence to the 15th December, and thence to the 24th January, met this day for Despatch of Business.
S3V0156P0_1	being seated on the Throne, and the Commons being at the Bar, with their Speaker, HER MAJESTY was pleased to make a most gracious Speech to both Houses of Parliament, as follows:—
S3V0156P0_2	in rising to move that an humble Address be presented by this House in answer to Her Majesty's most gracious Speech, said:—
S3V0156P0_3	My Lords, it is with diffidence that I rise to address your Lordships for the first time, and the greater is that diffidence when I recollect that although for many years I had the honour of a seat in the other House of Parliament, I remained throughout that time a silent actor in all the scenes to which my duties called me; but I will be brief, and the observations which I make shall be few, and I trust you will not weigh in too nice a balance any expressions which may fall from me on this occasion.
S3V0156P0_4	We hear from Her gracious Majesty that it is with gratification

is, the full Hansard debates. As we had previously discussed, this process is iterative, emphasizing that researchers transition from data processing to counting words to applying metrics on those word counts to visualizing trends, and then repeating this deductive process. Throughout this process, it may prove valuable to inspect and re-evaluate the intermediary steps that were taken to arrive at any conclusions about the corpus. This cyclical process may lead to additional historical questions and associations, which may mean more measuring, more visualization, and more guided reading in pursuit of a historical analysis.

Joining Debate Metadata

We have previously seen how to load the text from a single decade of the Hansard debates. But what should we do if we want to work with not only the text, but also important information like the name of the speaker who gave the speech and the date on which the speech was given?

First we can load the text data from the 1860s speeches, recalling that one of the columns, `sentence_id`, refers to the number of each sentence and its location in the nineteenth-century printed volumes of Hansard's parliamentary debates, referenced by series ("S"), volume ("V"), and page ("P"). Sentences are listed in the order in which they appeared on the page.

```r
# Use the hansardr library to load the debate text
library("hansardr")

data("hansard_1860")
```

Next, we can load the metadata corresponding with the Hansard debate text.

Table 2: Hansard Metadata 1860.

sentence_id	speechdate	debate
S3V0156P0_0	1860-01-24	MEETING OF THE PARLIAMENT.
S3V0156P0_1	1860-01-24	THE QUEEN'S SPEECH.
S3V0156P0_2	1860-01-24	ADDRESS IN ANSWER TO HER MAJESTY'S SPEECH.
S3V0156P0_3	1860-01-24	ADDRESS IN ANSWER TO HER MAJESTY'S SPEECH.
S3V0156P0_4	1860-01-24	ADDRESS IN ANSWER TO HER MAJESTY'S SPEECH.

Table 3: Hansard Metadata 1860.

sentence_id	speaker	suggested_speaker	ambiguous	fuzzy_matched	ignored
S3V0156P0_0	THE PARLIAMENT,		0	0	0
S3V0156P0_1	THE QUEEN	peter_quinn_4670	0	1	0
S3V0156P0_2	EARL FITZWILLIAM,	charles_fitzwilliam_1441	0	0	0
S3V0156P0_3	EARL FITZWILLIAM,	charles_fitzwilliam_1441	0	0	0
S3V0156P0_4	EARL FITZWILLIAM,	charles_fitzwilliam_1441	0	0	0

```r
data("debate_metadata_1860")
```

`hansardr` includes detailed contextual information for each of these speeches, using the same system of references that we saw in the previous data set – series, volume, page, and sentence number. In this dataset, however, there is no "text" column, but rather a series of columns that illuminate multiple "fields" of each sentence: the speech date (recorded as "speechdate" within the data) for when the speech was given and the official title of the "debate" in which the sentence was spoken.

In addition to the debate metadata, `hansardr` provides two additional metadata files: `speaker_metadata_1860` and `file_metadata_1860`.

```r
# Load Hansard speaker metadata for 1860
data("speaker_metadata_1860")
```

The `speaker_metadata_1860` dataset provides information about the "speaker" of each sentence as the MP's name was recorded in the original Hansard debates. It also includes a "suggested speaker" column, which contains a disambiguated version of the speaker names where ambiguous or inconsistent references to the same individual have been resolved into a single, standardized name, allowing for more accurate identification of MPs across the corpus.

The "suggested speakers" came from a large-scale disambiguation effort where the authors of this book disambiguated speaker names, building upon the work completed by the Digging into Linked Parliamentary Data Project and the Egger and Spirling database (Eggers and Spirling 2014; Winters and Steer 2013). This effort was designed to address these problems: speakers may have been called only by temporary office titles (like "Prime Minister"); speakers may have been called only by their peerage titles (such as "Duke" or "Viscount");a single name might have several permutations (such as W. Gladstone, W. E. Gladstone, or William Gladstone all referring to the same Mr. William Ewart Gladstone); speakers may have shared the exact name

Table 4: Source File Metadata 1860.

sentence_id	speech_id	debate_id	src_file_id	src_image	src_column
S3V0156P0_0	258488	30416	S3V0156P0	S3V0156P0I0033	1
S3V0156P0_1	258489	30417	S3V0156P0	S3V0156P0I0033	1
S3V0156P0_2	258490	30418	S3V0156P0	S3V0156P0I0036	7
S3V0156P0_3	258490	30418	S3V0156P0	S3V0156P0I0036	7
S3V0156P0_4	258490	30418	S3V0156P0	S3V0156P0I0036	7

with someone else who was active during the same period (like the two different Sir Robert Peels in the early 19th-century); speaker names may contain OCR errors; speaker names might have spelling inconsistencies due to how they were originally recorded during the 19th-century; some speakers were misrecorded, such as they were called by the wrong title.

By providing both the original speaker field as well as the disambiguated version, this dataset preserves the original historical recording and draws attention to the data transformation that occurred in the speaker field before the corpus was packed into `hansardr`. The purpose of including the "suggested speakers" field is to support computational analysis by linking variant or ambiguous names to consistent, canonical speaker identities for a more reliable analysis.

However, we also advise discretion. While disambiguation significantly improves analysis by standardizing speaker identities, it is not without limitations. In some cases, speakers may be incorrectly matched or mislabeled by the algorithm. In others, no suggested speaker name is provided because the necessary metadata was unavailable, and the sheer number of speakers made manual correction infeasible within our project timeline. To address these challenges, we have uploaded the Hansard corpus to Democracy Viewer, an online text mining and data sharing platform where researchers can also submit corrections to the corpus (Buongiorno et al., 2023). Over time, we plan to incorporate these contributions into future releases of the `hansardr` package.

Additionally, disambiguation can obscure the nuances of the original records, such as these shifts in titles or the historical importance of misspellings for understanding contemporary spelling conventions or clerical practices. For this reason, we provide both the original speaker field and the disambiguated version, so that researchers can critically assess and, if desired, use the original speaker name data instead.

Lastly, `hansardr` provides file metadata. For example, the `file_metadata_1860` dataset provides metadata about the records themselves, including the ID for the original source file, and the column in which the text appears.

```r
# Load Hansard file metadata for 1860
data("file_metadata_1860")
```

We have chosen to store the debate text (`hansard_1860`) separately from the contextual information about each speech (for example, `debate_metadata_1860` or `speaker_metadata_1860`), as well as the information about each hard copy file containing the speeches (`file_metadata_1860`). We subsetted the data into these individual files because the data for each decade is quite large. Processing such a large dataset can exceed a computer's resources, such as its memory. In other cases, we chose to exclude certain fields from processing if they were not immediately relevant to our analysis, as doing so reduces memory usage and improves overall processing speed so that we can focus on our research instead of waiting too long for any intermediary processing step to complete. Furthermore, many operations in this book – for instance, counting words per decade – require just a few datasets loaded at a time.

As a result of this organization scheme, many kinds of analyses will require working with multiple datasets from `hansardr` joined together. We will use the function `left_join()` to join the annotated information about speech context with each of the speech text. A left join returns every record from the data frame specified by the left-most argument, and all the matched records from the data frame on the right.

```
# Load the tidyverse
library("tidyverse")

# Join the Hansard debate text with the debate metadata into a single data frame
# left_join() will automatically detect that both datasets share the sentence_id
hansard_1860 <- left_join(hansard_1860, debate_metadata_1860)

# Show the first few rows of the 1860 Hansard debate text
head(hansard_1860)
```

```
##   sentence_id
## 1 S3V0156P0_0
## 2 S3V0156P0_1
## 3 S3V0156P0_2
## 4 S3V0156P0_3
## 5 S3V0156P0_4
## 6 S3V0156P0_5
##
## 1
## 2
## 3
## 4 My Lords, it is with diffidence that I rise to address your Lordships for the first time, and the
## 5
## 6
##   speechdate                                      debate
## 1 1860-01-24                  MEETING OF THE PARLIAMENT.
## 2 1860-01-24                         THE QUEEN’S SPEECH.
## 3 1860-01-24 ADDRESS IN ANSWER TO HER MAJESTY’S SPEECH.
## 4 1860-01-24 ADDRESS IN ANSWER TO HER MAJESTY’S SPEECH.
## 5 1860-01-24 ADDRESS IN ANSWER TO HER MAJESTY’S SPEECH.
## 6 1860-01-24 ADDRESS IN ANSWER TO HER MAJESTY’S SPEECH.
```

Notice that the resulting dataset, `hansard_corpus_1860`, has information about speaker and date for each sentence of text.

Joins are a crucial concept for combining and analyzing data that is distributed across multiple datasets. They allow us to link related information from different tables based on a shared key or common field thus enabling us to perform analyses of the records in relation to its metadata.

In the example above, both `hansard_1860` and `debate_metadata_1860` share a column called `sentence_id`. This shared column acts as a common identifier, enabling us to link the speaker and date information from one dataset to the corresponding text of the speech in the other dataset. By using this common identifier, the two datasets can be "joined" together to create a single, unified dataset that we will use for text processing.

Thus joins are so important because it is through linking the records to their metadata that we are able to break text into subsets and analyze change over time, or create a biography of any individual speaker. We can explore the words invoked in parliament in the months leading up to August 15, 1867, when most working-class men in Britain finally achieved the right to vote.

Multiword Phrases

Beyond analyzing individual words, an analyst may want to explore multiword phrases. Multiword phrases contain formulations of concepts that had powerful political, social, and cultural meaning to speakers, for example, "the will of the people," a phrase that has been invoked by both radicals and conservatives to support extremely different imaginations of the people. That is, speakers who invoked "the will of the people" might have been bolstering the importance of their speech by gesturing towards their status as a representative of the workers whose labor is the source of all political legitimacy. On the other hand, some speakers might have used the same phrase – "the will of the people" – to kindle an imagination of parliament as a zone a traditional folk whose national identity legitimizes a politics of excluding from citizenship or rights people with certain racial or immigration status. The question becomes: which version of the people was being represented in the discourse in question?

The analyst cannot anticipate in advance how such a phrase might have been used. The only way of gaining knowledge from a count of phrases is to "validate" one theory or the other by gaining more information.

We could, in theory, contrive computational approaches to help us discern why people invoked "the will of the people." But in *Text Mining for Historical Analysis*, we often want to move from identifying a data-driven phenomenon to gaining more information about historical context by reading the actual speeches. This approach provides insight into the rich nuances suggested by speakers by giving us perspective into how they invoke particular phrase in the original records. This is because, while visualizations might demonstrate that a parliamentarian invoked a phrase, they may not clearly demonstrate how that phrase was invoked.

Accordingly, in the sections below, we will begin by counting phrases, but that is not where we will end. We will return to the original speeches to read more deeply.

Finding Bigrams

While there are several approaches to finding multiword phrases, we will demonstrate this by tokenizing the text into bigrams.

The following code filters the speeches of the 1860s for the first months of 1867 before the vote on the Second Reform Act, which gave Britain's urban working-class men the right to vote. We also introduce a new package, `lubridate`, which can be installed in the usual way using `install.packages()`.

```r
# Install lubridate
install.packages("lubridate")
```

```r
# load packages for tokenization and date handling
library("tidytext")
library("lubridate")
```

```r
# create a new data frame "hansard_1867" from "hansard_1860" that contains
# data for 1867
hansard_1867 <- hansard_1860 %>% # create a new data frame
  mutate(year = year(speechdate)) %>% # extract year from speechdate
  filter(year == 1867, # keep only year 1867
         speechdate <= ymd("1867-08-15")) # and speechdate before or on august 15
```

Now that we have filtered the dataset for our desired timeline, we can tokenize the text into bigrams. The order in which we perform data processing is important. Tokenizing after filtering ensures that we are working with a smaller and more focused dataset. Working with a smaller dataset ensures we do not exhaust our computer's resources or its processing capabilities. This issue we think is paramount to processing large corpora for historical analysis, as we describe in more detail in subsequent chapters.

As a result, the order of these steps is not arbitrary, but the outcome of a deliberate decision. We could have reversed them—for example, tokenizing the text before filtering by date. However, that approach would have used more of our computer's resources, as it would require processing a larger volume of irrelevant text before narrowing the records down to just the data needed for our inquiry into a given time period. With large datasets like Hansard, the sequence of processing steps can significantly impact whether the data remains manageable.

In turn, the limitations of any computing environment directly shape the types of analyses we prioritize. In other words, it is not only theoretical or interpretive concerns that determine our methodological choices, but also what is computationally feasible. In this sense, technical constraints become analytical constraints.

When working with large-scale historical data, researchers often adapt their questions, models, and scope to align with the practical boundaries of a computer's available memory, processing speed, and storage. These limitations might encourage researchers to think critically and strategically about what kinds of knowledge are worth pursuing, and how best to pursue them with the tools at hand.

Throughout *Text Mining for Historical Analysis* we have attempted to limit barriers to processing a dataset as large as Hansard through our data organization scheme that breaks each decade of text and related metadata files into their own files, and presents readers with an order of intermediary data processing that is more efficient for working with historical data in R.

The following code uses the `unnest_tokens()` function, which we previously used to split text into individual words. This time, we adjust the function to extract pairs of consecutive words instead. To do this, we set the token argument to n-grams, instructing R to create sequences of consecutive words, and the `n` argument to 2, specifying that each sequence should consist of two words. These two arguments work together to extract bigrams.

```r
# Generate bigrams (two-word sequences) from the "text" column of the hansard_1867 data
bigrams_1867 <- hansard_1867 %>% # create a new dataset
  unnest_tokens(bigram, text, token = "ngrams", n = 2) # unnest bigrams
```

Note that the bigrams extracted overlap, as if the same sentence had been subdivided several times. This output is expected bigrams are created by looking at each pair of words in order, one step at a time—so each new bigram shares a word with the one before it.

Table 5: Hansard Bigrams with Metadata.

sentence_id	speechdate	debate	year	bigram
S3V0185P0_0	1867-02-05	THE QUEEN'S SPEECH.	1867	being seated
S3V0185P0_0	1867-02-05	THE QUEEN'S SPEECH.	1867	seated on
S3V0185P0_0	1867-02-05	THE QUEEN'S SPEECH.	1867	on the
S3V0185P0_0	1867-02-05	THE QUEEN'S SPEECH.	1867	the throne
S3V0185P0_0	1867-02-05	THE QUEEN'S SPEECH.	1867	throne adorned

Table 6: Clean Hansard Bigrams with Metadata.

sentence_id	speechdate	debate	year	bigram
S3V0185P0_0	1867-02-05	THE QUEEN'S SPEECH.	1867	throne adorned
S3V0185P0_0	1867-02-05	THE QUEEN'S SPEECH.	1867	regal ornaments
S3V0185P0_1	1867-02-05	THE QUEEN'S SPEECH.	1867	robes sitting
S3V0185P0_2	1867-02-05	THE QUEEN'S SPEECH.	1867	robes commanded
S3V0185P0_2	1867-02-05	THE QUEEN'S SPEECH.	1867	gentleman usher

Counting Bigrams

Now we can count the top bigrams and remove any that contain stop words. We remove stop words to focus on word pairs that might be more meaningful for our analysis.

```r
# remove bigrams that contain any stop words
# keep only bigrams made up of two lowercase alphabetic words
clean_bigrams_1867 <- bigrams_1867 %>%
  filter(!str_detect(bigram,
                     paste0("\\b(", paste(stop_words$word, collapse = "|"), ")\\b"))) %>%
  filter(str_detect(bigram, "^[a-z]+ [a-z]+$")) # keep only two-word lowercase alphabetic bigrams
```

The above code processes the dataset by applying the following transformations and filters:

`filter(!str_detect(...))` - Removes rows where the `bigram` column contains any stop word. - It constructs a regular expression from a list of stop words (`stop_words$word`) by: - Combining them with the OR operator (`|`). - Surrounding them with word boundaries (`\\b`) to match complete words only. - This ensures that only bigrams without stop words are retained.

`filter(str_detect(bigram, "^[a-z]+ [a-z]+$"))` - Keeps rows where the `bigram` column matches a specific pattern: - The `bigram` must consist of exactly two lowercase alphabetic words separated by a single space. - Rows where the `bigram` contains numbers, punctuation, or non-lowercase letters are excluded.

We can now count the resulting bigrams.

```r
# count the frequency of each bigram in the cleaned dataset
# keep the top 100 most frequent bigrams and sort them in descending order
top_bigrams_1867 <- clean_bigrams_1867 %>%
```

Table 7: Top Hansard Bigrams for 1867.

bigram	n
noble lord	1807
noble earl	1538
noble friend	1027
roman catholic	943
reform bill	742

```
  count(bigram) %>% # count frequency of each bigram
  top_n(100) %>% # select top 100 bigrams
  arrange(desc(n)) # sort bigrams by frequency, highest first
```

```
## Selecting by n
```

In this list, we see several contemporary rhetorical figures that stem from the practice of referring to other speakers in parliament as "my noble friend" or "my learned friend." We also see allusions to the debates over representation and the question of who would have the vote, especially in the phrases "household suffrage," referencing historical argument for granting middle-class men the right to vote with the Reform Bill of 1832, and "roman catholic," a reference to the fact that Roman Catholics could already vote until the Reform Act of 1834. Only in 1867 did those working-class men get the vote in Britain, and these earlier precedents were frequently alluded to as a justification.

We also see a reference to "public meetings." Why did speakers in parliament invoke public meetings in the lead-up to the Second Reform Act? We will answer this question by reading the original text.

```
# Filter the 1867 Hansard speeches to keep only those that mention "public meetings"
my_reading <- hansard_1867 %>% # create a new dataset
  filter(str_detect(text, "public meetings")) # detect the phrase

head(my_reading$text)
```

```
## [1] "I am aware that many of the public meetings have declared in favour of manhood suffrage, but ha
## [2] "They are often drawn up with considerable ability; but they bear the mark, I think, of a single
## [3] "Speeches at public meetings, and the discussions of the press, great as their influence undoubt
## [4] "I know I shall be asked, from what I have seen in the press, and what has been said at public m
## [5] "Have you not heard or read what has been said at public meetings of every kind?"
## [6] "There have been more than 1, 000 public meetings; at every one, the doors were open, and any ma
```

A cursory reading of the sentences suggests that members of parliament were well aware of massive public meetings where more than a thousand members of the public showed up support the vote for working-class people. We read of demonstrations around British Empire, even in Nova Scotia. We read of members of parliament challenging each other to meet the demands of the public: "Have you not heard or read what has been said?" We have only skimmed a few sentences, but the material in these 126 rows gives us what we need to understand how members of parliament talked about public protests in these crucial months of 1867.

Bigrams, however, have limitations. A bigram only looks at one preceding word, but meaning often depends on words far apart in a sentence. Observing longer utterances—such as by moving beyond bigrams to multiword phrases—allows us to capture linguistic meaning that depends on more distant context than just the immediately preceding word.

Counting Multiword Phrases

Text mining also allows us to look for multiword phrases beyond bigrams. We can look at 5-word phrases from the year 1867. We can also add a line to filter the n-grams for the presence of the word "people," looking for voices that refer to democracy, for instance, "the will of the people," an important concept in the years leading up to 1867, when working-class men in cities got the right to vote for the first time. The results will allow us to ask the question: What are the most frequently-invoked phrases involving "the people"?

We can use a slight adjustment of the code we used above to count bigrams to find n-grams of any length. Here is code for finding common five-word-phrases, filtering them, counting them, and finding the top 50 examples:

```r
# create 5-grams (sequences of 5 consecutive words) from the "text" column
# store the results in a new column called "ngram"
# unnest_tokens() is used with token = "ngrams" and n = 5 to generate 5-word sequences
fivegrams_1867 <- hansard_1867 %>% # create a new dataset
  unnest_tokens(ngram, text, token = "ngrams", n = 5) # generate 5-grams from text

# filter for 5-grams that contain the word "people"
# count the frequency of those 5-grams and keep the top 50
people_n_grams <- fivegrams_1867 %>% # create a new dataset
  filter(str_detect(ngram, "people")) %>% # keep 5-grams containing "people"
  count(ngram) %>% # count frequency of each 5-gram
  top_n(50) # keep top 50 by count
```

```
## Selecting by n
```

```r
head(people_n_grams)
```

```
##                           ngram  n
## 1    feelings of the people of  8
## 2     great body of the people 32
## 3 great majority of the people 13
## 4     great mass of the people 15
## 5 majority of the irish people 10
## 6    majority of the people of 27
```

We see a great many political concepts invoked about what members of parliament believed they were working for – the people of Ireland, England, or the metropolis, their representation, condition, education, minds, interests, feelings, rights, and welfare. An entire vocabulary of social engagement and political will was being spelled out.

Reading further down this list might give us more hints about the language with which speakers urged the principle of representation. Again, the careful analyst would trace these words back to their original context, using the original text of the speeches to develop an argument about why a phrase like “the minds of the people” was used.

```r
# Filter the 1867 Hansard speeches to include only rows where the exact phrase
# "minds of the people" appears in the "text" column
minds <- hansard_1867 %>% # create a new dataset
  filter(str_detect(text, "minds of the people")) # detect the phrase

head(minds$text)
```

```
## [1] "It is satisfactory to learn that these measures afforded great relief to the minds of the people
## [2] "Its debates have been the principal means by which political wisdom, and the results arrived at
## [3] "Its debates command nothing like the same interest and attention, and exercise far less influenc
## [4] "Gentleman the Question of which he had given notice, and was sure the Government would sacrifice
## [5] "He believed, however, that the partial administration of justice in Ireland sometimes created a
## [6] "The question is whether the impression is or is not to be conveyed to the minds of the people o
```

When speakers referenced the “minds of the people,” they frequently depicted popular politics as a form of collective political achievement to which ordinary people contributed the rational evidence of their own lived experience. We find the phrase connected to ideas about moral “duty” and political “harmony.” We also see it invoked in reference to Ireland, India and Scotland.

These images of the peoples’ collective wisdom contrast against the bias expressed by elites in another age, which warned against democracy as a form of potential despotism governed by ignorance. It is not surprising that the months leading into the Reform Act of 1867 would see eulogies to the wisdom of popular politics, but the phrase “the minds of the people” offers one entry-point for analysis.

Using Bigrams to Find Mentions of Events

For many kinds of textual analyses, incorporating external information can help guide the focus of the analysis and ensure it aligns with specific research questions. For example, a historian analyzing data might be interested in tracking references to notable events in British history, such as the Magna Carta, the Spanish Inquisition, or the Glorious Revolution, to see how often these events are mentioned in parliamentary records. To achieve this, the analyst may create and use a “controlled vocabulary” which is a predefined list of terms or topics of interest related to the analysis. `hansardr` includes an example controlled vocabulary: a curated list of significant events in British history, along with the corresponding years of their occurrence.

In applying this list, this exercise invites readers to investigate the idea that representations of memory change over time using the controlled vocabulary to search for a set group of phrases. We will use a controlled vocabulary to measure Parliamentarians’ references to past events in the 1860 debates. We are intentionally choosing words such as “riot” or “meeting” that might collect evidence of parliamentary speakers referencing public meetings like those we saw evidence of above. We are also curious about how those references compare with references to famines, wars, strikes, exhibitions, and other contemporary and historical events.

```
# Load the Hansard speech data and the corresponding debate metadata for the 1860s
data("hansard_1860")
data("debate_metadata_1860")

# merge the hansard speech data with its metadata using a left join
# this retains all rows from hansard_1860 and adds matching metadata columns
hansard_1860 <- left_join(hansard_1860, debate_metadata_1860)
```

```
## Joining with ‘by = join_by(sentence_id)‘
```

```
# create a new dataset for just the year 1867
hansard_1867 <- hansard_1860 %>% # create a new data frame
  mutate(year = year(speechdate)) %>% # keep only speeches from the year 1867,
  filter(year == 1867, #keep only speeches made on or before August 15th
         speechdate <= as.Date("1867-08-15"))
```

```
# Create bigrams (2-word sequences) from the "text" column of the 1867 Hansard data
# - "bigram" is the name of the new column to store the 2-word phrases
# - "text" is the input column containing the speech text
# - token = "ngrams" with n = 2 extracts bigrams
bigrams_1867 <- hansard_1867 %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

# remove bigrams that contain any stop word (e.g., "the", "of", "and")
# and keep only bigrams made up of two lowercase alphabetic words (exclude numbers and punctuation)
clean_bigrams_1867 <- bigrams_1867 %>%
  filter(!str_detect(bigram, paste0("\\b(",
                                    paste(stop_words$word, collapse = "|"), ")\\b"))) %>%
  filter(str_detect(bigram, "^[a-z]+ [a-z]+$"))

head(clean_bigrams_1867)
```

```
##   sentence_id speechdate              debate year          bigram
## 1 S3V0185P0_0 1867-02-05 THE QUEEN’S SPEECH. 1867  throne adorned
## 2 S3V0185P0_0 1867-02-05 THE QUEEN’S SPEECH. 1867 regal ornaments
## 3 S3V0185P0_1 1867-02-05 THE QUEEN’S SPEECH. 1867   robes sitting
## 4 S3V0185P0_2 1867-02-05 THE QUEEN’S SPEECH. 1867 robes commanded
## 5 S3V0185P0_2 1867-02-05 THE QUEEN’S SPEECH. 1867 gentleman usher
## 6 S3V0185P0_2 1867-02-05 THE QUEEN’S SPEECH. 1867       black rod
```

The following code filters the `clean_bigrams_1867` dataset to retain only rows where the bigram column contains any word from the pattern1 list (e.g., “riot,” “meeting,” “famine”). It uses a regular expression to match whole words, ensuring accurate filtering. After filtering, the resulting dataset, `events_1867`, includes only the bigram and speechdate columns. This allows the analyst to focus on specific bigrams related to the predefined topics of interest.

```
# define a controlled vocabulary of event-related keywords to search for in bigrams
controlled_vocab = c("riot", "meeting", "famine","revolt", "exhibition", "massacre",
                     "strike", "war")

# filter the cleaned bigrams to keep only those that contain one of the vocabulary terms
# build a regex pattern from the vocabulary and match bigrams containing any of those words
# select only the bigram and the associated speech date for further analysis
events_1867 <- clean_bigrams_1867 %>%
  filter(str_detect(bigram,
                    paste0("\\b(", paste(controlled_vocab, collapse = "|"), ")\\b"))) %>%
  select(bigram, speechdate)

head(events_1867)
```

```
##             bigram speechdate
## 1  sanguinary war 1867-02-05
## 2 dreadful famine 1867-02-05
## 3        late war 1867-02-05
## 4       civil war 1867-02-05
## 5       civil war 1867-02-05
## 6       china war 1867-02-08
```

If we look carefully at the events list so generated, we will recognize only a few of these references as actual events. Many of the bigrams including our controlled vocabulary are simply descriptions, for example, “disorderly riot.” A few are references to actual events and meetings specified by the name of an event, for instance, the “Bristol Riot” or a “Yorkshire meeting.” Far more frequent are general references to a “public meeting” or “recent meeting.” All of this is interesting, of course, because as we have seen members of parliament leaned on each other to acknowledge the public sentiment for expanding the vote.

How did references to meetings vary over time? By this time, you will have noticed that searching for a controlled vocabulary, visualization, guided reading, and deliberation about the meaning of the data is an iterative process that we engage over many rounds – not an automatic process where the analyst looks for a word like “riot” and immediately finds meaningful results. We can search for our key phrases and visualize them.

In the following code, we define a new list of bigrams that make up our controlled vocabulary—specific phrases referring to historical events—and then filter the dataset to keep only the rows where these phrases appear. Note that the code below uses a new command for faceted counting–`group_by()`–which we will investigate in greater detail throughout the book.

For now, it is just important to know that `group_by()` effectively sorts data into buckets based on some category. For example, you could use group_by() to organize the data by year—putting all the speeches from the same year into one group—or by speaker, grouping together everything each person said. By grouping, we are able to perform calculations on each category individually. Instead of counting words for an entire data subset, like we did in Chapter 1, we can count words by group. In the following code we use `group_by()` to combine and count each bigram.

```
# define a controlled vocabulary of specific multi-word historical events (as bigrams)
controlled_vocab_events = c("yorkshire meeting", "clontarf meeting", "recent meeting",
                            "public meeting", "paris exhibition", "orissa famine",
                            "irish famine", "crimean war", "civil war", "affghan war",
                            "kaffir war", "bristol riot")

# filter the events to keep only those that exactly match a defined historical event
# group the matched events and count their occurrences
# keep only the top 10 most frequently mentioned events
top_events_1867 <- events_1867 %>%
  filter(bigram %in% controlled_vocab_events) %>% # match exact event bigrams
  group_by(bigram) %>% # group by event phrase
  summarize(total = n()) %>% # count occurrences
  top_n(10) # keep top 10 events

head(top_events_1867)
```

```
## # A tibble: 6 x 2
##   bigram            total
##   <chr>             <int>
## 1 affghan war           3
## 2 bristol riot          1
## 3 civil war            31
## 4 clontarf meeting      1
## 5 crimean war          55
## 6 irish famine          5
```

```
# join the top event bigrams with the full events dataset to recover their speech dates
# extract the month, count mentions by date, and return a regular data frame
top_events_1867_w_speech_metadata <- top_events_1867 %>%
  left_join(events_1867, by = "bigram") %>%  # join to recover speech dates
  mutate(month = month(speechdate)) %>% # extract month from speechdate
  count(bigram, speechdate) %>% # count mentions by date
  ungroup() # remove grouping

head(top_events_1867_w_speech_metadata)
```

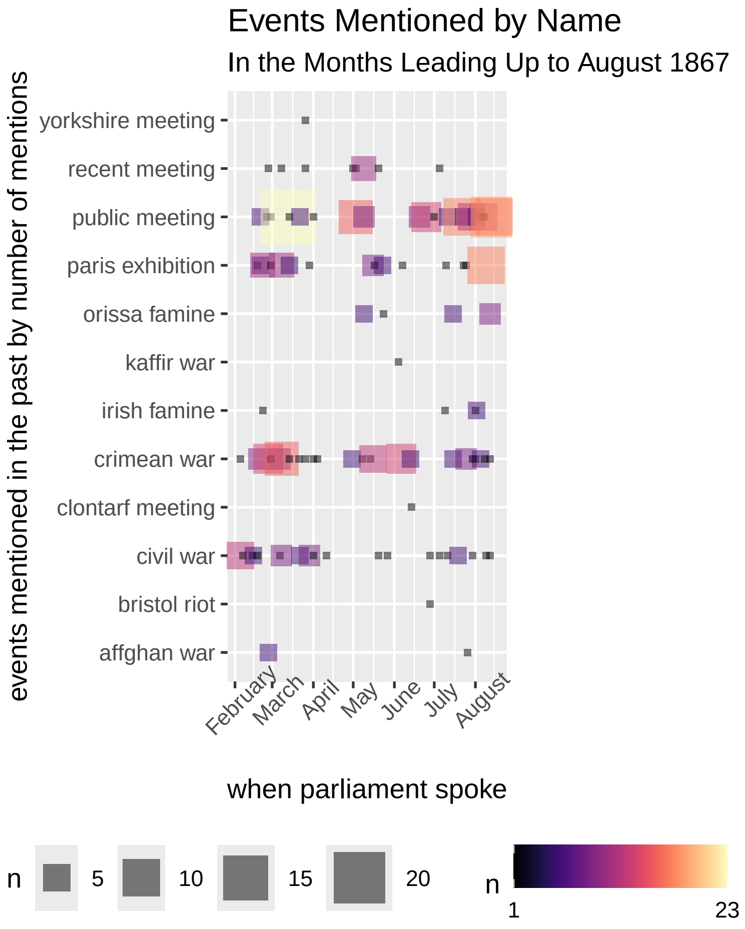
```
## # A tibble: 6 x 3
##   bigram       speechdate     n
##   <chr>        <IDate>    <int>
## 1 affghan war  1867-02-26     2
## 2 affghan war  1867-07-26     1
## 3 bristol riot 1867-06-28     1
## 4 civil war    1867-02-05     5
## 5 civil war    1867-02-07     1
## 6 civil war    1867-02-14     1
```

```r
install.packages("viridis")
```

```r
# Load the viridis color palette library for colorblind-friendly colors
library("viridis")
```

```
## Loading required package: viridisLite
```

```r
# Create a bubble plot showing how often specific events were mentioned over time
ggplot(top_events_1867_w_speech_metadata,
       aes(x = speechdate, # x-axis: date of the speech
           y = bigram, # y-axis: event bigram (e.g., "irish famine")
           size = n, # bubble size represents number of mentions
           color = n)) + # bubble color also reflects number of mentions

  # Use a viridis color scale with a log transformation for good contrast in frequency
  # treat "n" as continuous, not categorical, for a smooth gradient of colors
  scale_color_viridis(breaks = round, # apply rounding to color scale breaks
                      trans = "log", # log-transform to spread small/large counts
                      option = "A", # use Viridis option A color scheme
                      discrete = FALSE, # treat "n" as continuous
                      direction = 1) + # default color direction (ascending)

  # format and customize the appearance of a timeline plot showing event mentions
  # x-axis shows months, point size reflects frequency, and plot is styled for readability
  # includes rotated labels, extended margins, and simplified legend
  scale_x_date(date_breaks = "1 month", date_labels = "%B") + # x axis one label per month
  scale_size_continuous(range = c(1, 10)) + # scale point sizes
  geom_point(alpha = .5, shape = 15) + # semi-transparent square points
  coord_cartesian(clip = "off") + # don't clip outer elements

  # appearance tweaks: legend, axis, margin, and label formatting
  theme(legend.position = "bottom", # legend below plot
        axis.text.x = element_text(angle = 45), # tilt x labels
        plot.margin = unit(c(1, 20, 1, 1), "lines")) + # widen right margin
  guides(shape = "none") + # hide shape legend

  # axis labels and titles
  labs(x = "when parliament spoke", # x label
       y = "events mentioned in the past by number of mentions", # y label
       subtitle = "In the Months Leading Up to August 1867") + # subtitle
  ggtitle("Events Mentioned by Name") # main title
```



What do we make of this timeline? It suggests a swell of public meetings referenced in parliament in the months leading up to August 1867, which received more intense attention than either the Crimean War or the Paris Exhibition. Individual meetings such as those in Yorkshire or Clontarf received passing acknowledgment rather than deeper debate. Understanding more than this requires more cycles of research and reading.

Other research questions appear that the individual analyst, engaged in a process of research, would want to follow. Did references to public meetings change from January to August, while their mentions intensified? Were all of these public meetings about the vote? What was the meaning of references to a “civil war” – the English civil war or recent events in America – and was this civil war invoked in reference to 1867? What about the references to the contemporary famine in Orissa (in India) or the (presumably historical) famine in Ireland?

One way to approach this line of inquiry would be by using the graph we have generated as a source of concrete questions about how members of parliament invoked current and historical events in the lead-up to the vote on the Second Reform Act. An analyst of history would follow up on most of these questions by reading individual sentences, speeches, and entire debates and to form these readings into a historical argument about the role of events in shaping political reform.

Finding Historical Events Using a Controlled Vocabulary

Next, let's return to the idea of finding events using what we have learned about joins. The authors of this book have created a controlled vocabulary of event names which lists the date of famous events like the Magna Carta (1215).

As *The Dangerous Art of Text Mining* makes clear, the authors of this book understand that there is no universal definition of the most important events in world history. Any controlled vocabulary list has its merits and its omissions. It is in comparing the subtleties of different lists of events that the historian begins to explore the changing political and cultural nature of memory – for example, the fact that references to the Glorious Revolution began to disappear after the First Reform Act, gradually displaced by allusions to the Tudor past.

```
# load the scholar-created "events" dataset from hansardr
data("events")

# process the dataset to get a cleaned list of historical events
# each row represents a unique event with its assigned historical date
# ensures lowercase names for matching and excludes events after 1840
eventslist <- events %>% # create a new dataset
  distinct(event_name, scholar_assigned_date) %>% # remove duplicates
  filter(!scholar_assigned_date > 1840) %>% # exclude post-1840 events
  mutate(event_name = tolower(event_name)) %>% # lowercase event names
  select(event_name, scholar_assigned_date) # keep only name and date

head(eventslist)
```

```
## # A tibble: 6 x 2
##   event_name          scholar_assigned_date
##   <chr>                               <dbl>
## 1 french revolution                    1789
## 2 magna carta                          1215
## 3 norman conquest                      1066
## 4 corn laws                            1815
## 5 battle of boyne                      1690
## 6 glorious revolution                  1688
```

Notice that events contains two columns: events by their name, and the historical dates on which those events occurred.

We will now count events mentioned in parliamentary speeches, starting with using an `inner_join()`. Like its cousin, `left_join()`, `inner_join()` works with two data sets to make a new, combined data set. We use `inner_join()` when we want to analyze shared areas of overlap between two data sets. In this case, we want to find the n-grams from `annotated_1830` that are also listed as entities in the events controlled vocabulary.

```
# Load the Hansard speech text and associated metadata for the year 1830
data("hansard_1830")
data("debate_metadata_1830")

# Merge the speech data with its metadata using a left join
# All rows from hansard_1830 are retained, with metadata added where available
hansard_1830 <- hansard_1830 %>% # create a new dataset
  left_join(debate_metadata_1830) # join debate text and metadata
```

```
## Joining with ‘by = join_by(sentence_id)‘
```

```
# Tokenize the speech text into bigrams
# - "ngram" is the name of the new column to store the bigrams
# - "text" is the input column containing the speech text
# - token = "ngrams" with n = 2 instructs the function to extract bigrams
bigrams_1830 <- hansard_1830 %>% # create a new dataset
  unnest_tokens(ngram, text, token = "ngrams", n = 2) # tokenize text column

# Identify bigrams that match known historical events from "eventslist"
# - Performs an inner join where the bigram matches the cleaned event name
# - Keeps only rows with a match, linking bigram use to historical context
events_1830 <- bigrams_1830 %>% # create a new dataset
  inner_join(eventslist, by = c("ngram" = "event_name")) # join the bigrams and events

head(events_1830)
```

```
##      sentence_id speechdate                                       debate
## 1  S2V0022P0_158 1830-02-04 ADDRESS ON THE LORDS COMMISSIONERS SPEECH. ]
## 2 S2V0022P0_1722 1830-02-18                                  POUTUGAL. ]
## 3 S2V0022P0_1729 1830-02-18                                  POUTUGAL. ]
## 4 S2V0022P0_1735 1830-02-18                                  POUTUGAL. ]
## 5 S2V0022P0_1833 1830-02-18                                  POUTUGAL. ]
## 6 S2V0022P0_1923 1830-02-18                                  POUTUGAL. ]
##                    ngram scholar_assigned_date
## 1         navigation laws                  1660
## 2 portuguese constitution                  1822
## 3 portuguese constitution                  1822
## 4 portuguese constitution                  1822
## 5 portuguese constitution                  1822
## 6 portuguese constitution                  1822
```

Notice that the resulting dataset contains speaker and date columns from the parliamentary speeches, but the only bigrams listed are those that correspond to events from `eventslist`.

In the next blocks of code, we use our new list of events to generate a visualization.

```
# count how many times each event bigram is mentioned on each speech date
# extract year from date, group by relevant fields, and remove grouping after summarizing
```

```
counted_events <- events_1830 %>% # create a new dataset
  mutate(year = year(speechdate)) %>% # extract year from speech date
  group_by(ngram, year, speechdate, scholar_assigned_date) %>% # group data
  summarize(n = n()) %>% # count mentions
  ungroup() # remove grouping
```

```
## `summarise()` has grouped output by 'ngram', 'year', 'speechdate'. You can
## override using the `.groups` argument.
```

```
head(counted_events)
```

```
## # A tibble: 6 x 5
##   ngram                  year speechdate scholar_assigned_date     n
##   <chr>                 <dbl> <IDate>                    <dbl> <int>
## 1 american constitution  1831 1831-10-05                  1776     1
## 2 american constitution  1831 1831-10-06                  1776     2
## 3 american constitution  1832 1832-04-13                  1776     2
## 4 belgian revolution     1831 1831-08-09                  1830     1
## 5 belgian revolution     1831 1831-08-18                  1830     1
## 6 belgian revolution     1832 1832-03-16                  1830    10
```

```
# summarize total mentions of each event across all speech dates
# grouped by event name and assigned historical date
# then sort chronologically for historical analysis
cleaned_events <- counted_events %>%
  group_by(ngram, scholar_assigned_date) %>% # group by event and historical date
  summarize(total = sum(n)) %>% # sum total mentions
  ungroup() %>% # remove grouping
  arrange(scholar_assigned_date) # sort by historical date
```

```
## `summarise()` has grouped output by 'ngram'. You can override using the
## `.groups` argument.
```

```
head(cleaned_events)
```

```
## # A tibble: 6 x 3
##   ngram              scholar_assigned_date total
##   <chr>                              <dbl> <int>
## 1 norman conquest                     1066     5
## 2 twelfth century                     1100     3
## 3 thirteenth century                  1200     7
## 4 fourteenth century                  1300     2
## 5 game laws                           1389   112
## 6 fifteenth century                   1400     7
```

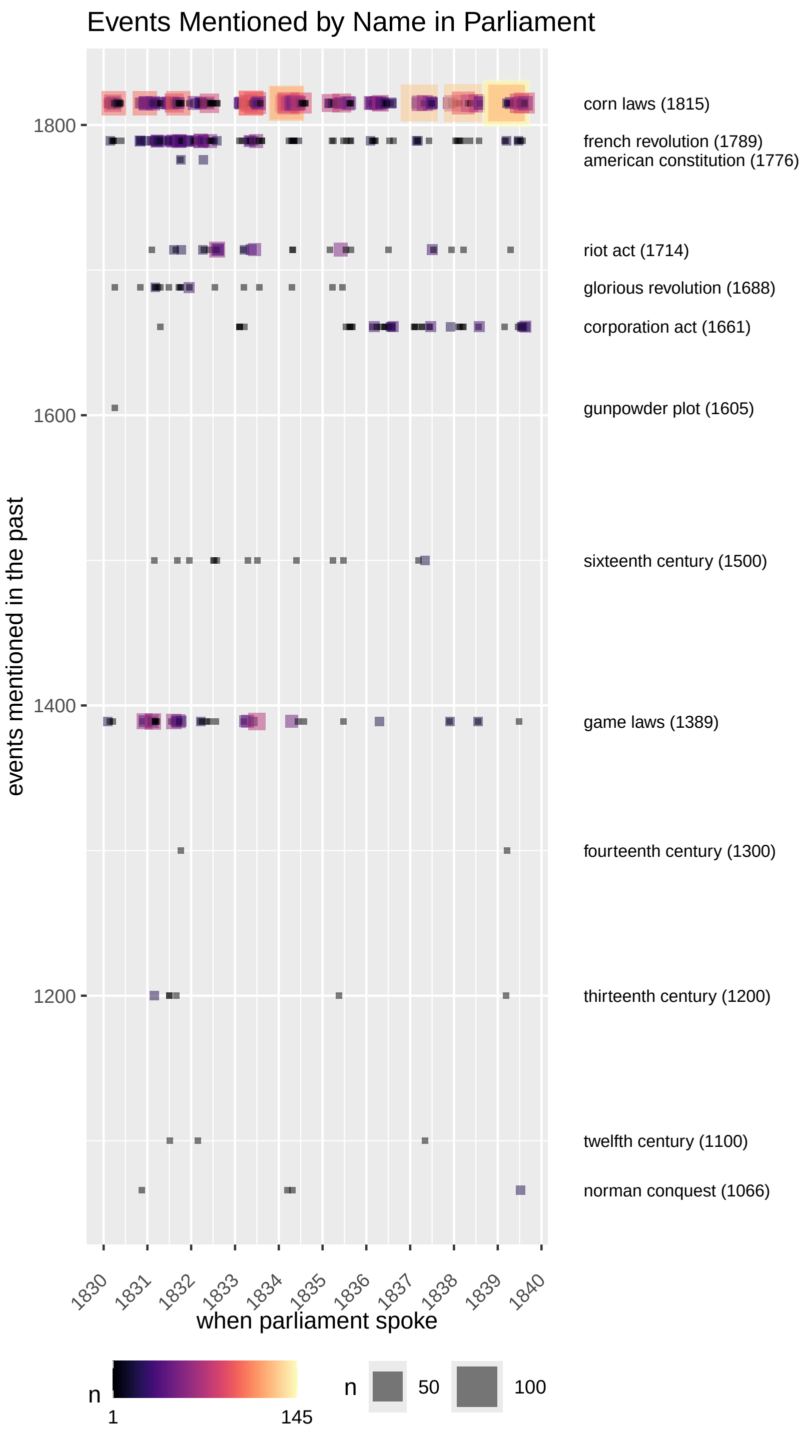
```r
# create an index to divide the historical date range into 30 equal chronological bins
# group by each bin and select the most frequently mentioned event within each
# then join back with the full event data to recover speech-level detail
indexed_events <- cleaned_events %>%
  mutate(index = scholar_assigned_date %/%
           ((max(scholar_assigned_date) -
               min(scholar_assigned_date)) / 30)) %>% # assign bin index
  group_by(index) %>% # group by time bin
  filter(total == max(total)) %>% # keep most mentioned event in each bin
  ungroup() %>% # remove grouping
  left_join(counted_events, by = c("ngram", "scholar_assigned_date")) # join data

head(indexed_events)
```

```
## # A tibble: 6 x 7
##   ngram           scholar_assigned_date total index  year speechdate     n
##   <chr>                           <dbl> <int> <dbl> <dbl> <IDate>    <int>
## 1 norman conquest                  1066     5    41  1830 1830-11-15     1
## 2 norman conquest                  1066     5    41  1834 1834-03-14     1
## 3 norman conquest                  1066     5    41  1834 1834-04-25     1
## 4 norman conquest                  1066     5    41  1839 1839-07-12     2
## 5 twelfth century                  1100     3    43  1831 1831-07-06     1
## 6 twelfth century                  1100     3    43  1832 1832-02-27     1
```

```r
# Define a boundary value by adding 8 days to the latest speech date in the dataset
# Make the visualization easier to read by adding padding to the visualization, ensuring no points are
left_range <- max(events_1830$speechdate) + 8
```

```r
# create a bubble plot showing when historical events were mentioned in parliament
# x-axis shows the date of mention; y-axis shows the historical date of the event
# bubble size and color represent frequency of mentions
ggplot(data = indexed_events, # provide the name of the dataset
       aes(x = speechdate, # assign the x-axis
           y = scholar_assigned_date, # assign the y-axis
           size = n, # assign the size of the points
           label = paste0("            ", ngram, " (", scholar_assigned_date, ")"),
           color = n)) +  # assign the color

# use viridis color scale with log transform for better contrast
# set bubble size aesthetics for visual clarity
# use viridis option A (purple to yellow-green gradient)
# and use a forward gradient (low values = dark, high = bright)
  scale_color_viridis(breaks = round, # round legend breaks
                      trans = "log", # log scale
                      option = "A", # viridis option
                      discrete = FALSE, # continuous scale
                      direction = 1) + # forward gradient
```

```r
# set up point appearance, plot boundaries, and x-axis formatting
# this controls how bubbles are sized, shaped, placed, and how time is shown
scale_size_continuous(range = c(1, 10)) + # bubble size range
geom_point(alpha = .5, shape = 15) + # square bubbles
coord_cartesian(clip = "off") + # no clipping of labels
scale_x_date(date_breaks = "1 year", date_labels = "%Y") + # yearly ticks on x-axis

# add one label per event, placed at the right edge and aligned with its historical date
geom_text(data = indexed_events %>% # assign data
          group_by(ngram) %>% # group data
          sample_n(1), # pick one speech instance per event
          show.legend = FALSE, # no text in legend
          aes(color = 1, # fixed color for all labels
              x = left_range, # position label at far right
              y = scholar_assigned_date, # align vertically with event date
              hjust = 0, # left-align label text
              size = 8)) + # label size

# style plot appearance for readability and layout
theme(legend.position = "bottom", # move legend below plot
      plot.margin = unit(c(1, 250, 50, 1), unit = "pt"), # add right margin for labels
      axis.text.x = element_text(angle = 45, vjust = 0.5, hjust = 1), # rotate x-axis labels
      axis.title.x = element_text(vjust = -0.5)) + # adjust x-axis title spacing
guides(shape = "none") + # remove shape legend

# add axis labels and title
labs(x = "when parliament spoke", # assign the x-axis
     y = "events mentioned in the past") + # assign the y-axis
ggtitle("Events Mentioned by Name in Parliament") # create a title
```


Events Mentioned by Name in Parliament
events mentioned in the past
1800
1600
1400
1200
corn laws (1815)
french revolution (1789)
american constitution (1776)
riot act (1714)
glorious revolution (1688)
corporation act (1661)
gunpowder plot (1605)
sixteenth century (1500)
game laws (1389)
fourteenth century (1300)
thirteenth century (1200)
twelfth century (1100)
norman conquest (1066)
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
when parliament spoke
n
1
145
n
50
100

Note that the visualization here resembles the visualization of events we generated above, but in this case, we have some extra information: the actual date of the event mentioned, e.g. 1789 for the French Revolution. We have plotted this historical information on the y-axis. The resulting graph in fact shows two timelines rather than one – a timeline of parliamentary speech on the x-axis and a timeline of historical references on the y-axis.

How should we interpret this graph? One approach is to identify the most surprising aspects of the visualization and explore their implications. For instance, as a British historian, I am unsurprised by frequent references in Parliament to events from the previous half-century, such as the French and American Revolutions. Similarly, it's expected that the Glorious Revolution was heavily referenced in the early 1830s, with mentions tapering off after 1836, and that the Riot Act was invoked annually. However, the most striking feature of the graph may be the vertical line around 1832, where references to much older historical periods—the sixteenth, thirteenth, and twelfth centuries—appear. This invocation of the distant past invites further investigation into its context and significance.

Why? I find myself asking. That verticality of temporal experience–the long perspective on the deep past–seems to be fairly distinct for 1830-32 of all the years in the decade. Why do so many different, varied periods suddenly become relevant all at once? Is this a general characteristic of moments of political reform – that the deep past is trawled for a multitude of examples, for and against?

We continue the explanation through reading.

```r
# add a new column extracting the year from the speech date
hansard_1830 <- hansard_1830 %>%
  mutate(year = year(speechdate))

# filter speeches from the year 1832 that mention the phrase "fifteenth century"
# - str_detect() searches for the exact phrase in the text
# - select() keeps only the "text" column of the matching speeches
c15_mentions <- hansard_1830 %>% # create a new dataset
  filter(year > 1831 & year < 1833, # filter for the date range
         str_detect(text, "fifteenth century")) %>% # detect the phrase
  select(text) # select just the text column
```

```r
c15_mentions$text
```

```
## [1] "The celebrated Act of William (1696) was only a transcript from a Scottish law of the fifteenth
```

```r
# Filter speeches from the year 1832 that mention the phrase "fourteenth century"
# - Only rows with "fourteenth century" in the text are retained
# - Only the "text" column is selected
c14_mentions <- hansard_1830 %>% # create a new dataset
  filter(year > 1831 & year < 1833, # filter for the date range
         str_detect(text, "fourteenth century")) %>% # detect the phrase
  select(text) # select just the text column
```

```r
# display the filtered speech texts that mention "fourteenth century"
c14_mentions$text
```

```
## character(0)
```

```
# Filter speeches from the year 1832 that mention the phrase "thirteenth century"
# - year > 1831 & year < 1833 captures only the year 1832
# - str_detect() looks for the exact phrase "thirteenth century" in the text
# - select() keeps only the "text" column of matching speeches
c13_mentions <- hansard_1830 %>% # create a new dataset
  filter(year > 1831 & year < 1833, # filter for the date range
         str_detect(text, "thirteenth century")) %>% # detect the phrase
  select(text) # select just the text column
```

```
# display the filtered speech texts that mention "thirteenth century"
c13_mentions$text
```

```
## character(0)
```

```
# filter speeches from the year 1832 that mention the phrase "twelfth century"
# - str_detect() searches for the exact phrase in the "text" column
# - select() keeps only the "text" column
c12_mentions <- hansard_1830 %>% # create a new dataset
  filter(year > 1831 & year < 1833, # filter for the date range
         str_detect(text, "twelfth century")) %>% # detect the phrase
  select(text) # select just the text column
```

```
# display the filtered speech texts that mention "twelfth century"
c12_mentions$text
```

```
## [1] "Why, there is not now a bricklayer who falls from a ladder in England, who cannot obtain surgica
```

What we see is a collection of historical references which serve two purposes: one is explaining why the institutions of the past differ from the needs of the present, and one is invoking the fact that parliamentary institutions have changed in the past to support the cause of reform in the present. The arguments work together, not against each other.

While we acknowledge that other historians have studied the discourse of parliamentary reform in detail, and even arrived at this same conclusion, what we find from a data-driven analysis of historical references to past events in parliament is the distinctiveness of 1832 as a moment of memory. Understanding how memory was used requires deep reading and a knowledge of the context of the debates over the vote; but understanding that historical memory was being leveraged in a particularly intensive way in 1832 is an insight that comes from treating text as data.

Data Processing Order

Throughout this chapter, we worked with large datasets and showed how to handle them effectively The process we demonstrated is especially important when handling a dataste as large the Hansard corpus. For example, if we try to tokenize the entire dataset before narrowing it down, the tokenization steps can exceed our computer's resources. Instead, by first filtering the dataset to include only the debates we care about–and then tokenizing–we can significantly reduce the processing load.

Here is a typical workflow for handling large data like Hansard:

- Load the Dataset: Import the dataset into your R environment using appropriate functions like `read_csv()`.
- Filter or Subset the Dataset: Narrow down the dataset to include only the data relevant to your analysis. For example, if we are analyzing speeches by William Gladstone, we can filter rows where the speaker is Gladstone using functions like `filter()` from the `dplyr` package. Similarly, we could filter for speeches belonging to a specific debate by filtering the data based on a title. Subsetting early in the process reduces the amount of data that will be processed later, saving processing time and computational resources.
- Clean the Dataset: After subsetting, clean the dataset to prepare it for analysis. Cleaning typically involves handling missing values, removing stop words, and normalizing text (e.g., converting text to lowercase). This can be done with functions like `mutate()` from `dplyr`, or string processing functions like `str_to_lower()`, which transforms text to lower case, from the `stringr` package.
- Analyze the Dataset: Now that you have just the needed data, perform your analysis by applying metrics.

Although it is poossible clean the entire dataset before narrowing it down, doing so can strain our computer's processing power or even exceed its available memory. It is often more desirable to filter the dataset first, then clean the smaller, more manageable portion. Following the order: Load -> Subset -> Clean -> Analyze can optimize data processing, which may be especially useful when analyzing change over decades or centuries of data.

Exercises

1) In the code above, we searched for 5-word phrases that reference "the people." Use the code to find two-word phrases in 1830, 1840, 1850, and 1860 that reference "meeting." Write a paragraph, quoting at least 10 phrases from each decade. For each of the phrases mentioned more than five times, describe what prejudices, positive or negative, the phrase encapsulates. Use your evidence to answer the question: how did parliamentary references to meetings change between 1830 and 1860?

Tips: Instead of joining every decade together, process the decades separately to avoiding using all your computer's resources.

2) In the code above, we created two kinds of timelines, one for 1867 using a controlled vocabulary of words such as "meeting" and the other for the 1830s using the list of historical events, `eventlist`. Make two more timelines. Apply the controlled vocabulary to the 1830s, and search for the items in `eventlist` in the 1860s. Use the code above to graph the results. Keep in mind that you will need to adjust your controlled vocabulary to what you find in the data.
3) Analyze the results of your timelines. Use iterative investigations of phrases, as modeled above, to conduct a distant reading of the parliamentary speeches. Using at least ten examples, compose a one-page essay investigating the question: how did parliamentary speakers use references to contemporary and historical events to make arguments for or against political change?

Practicing with AI

Throughout this book, we emphasize the importance of an iterative approach to analysis. During this iterative process we can refine one's own understanding of the data by revisiting the original records, or by questioning one's assumptions of historical interpretation through interrogative processes that build upon one-another.

AI chatbots can support iterative research, too, by supporting an interactive bnrainstorming process. Analysts can ask questions, review responses, and adjust their queries, engaging in a cycle of exploration and refinement. When used this way, an AI chatbot is guided by the analyst to generate focused, relevant responses. The analyst must also critique the chatbot's output to ensure it is accurate and useful. This approach allows analysts to use AI chatbots as a tool for inquiry while maintaining control over the research process.

In the following example, we use our queries from earlier in the chapter to guide an AI chatbot in helping us to brainstorm additional events that we might want to explore. Specifically, we ask the chatbot to generate additional events that might be of interest to someone studying language relating to war in the 1860 Hansard debates. We will guide the chatbot's response by giving it an example of the types of events that interest us taken from the `eventslist` dataset.

The template we provide the chatbot (e.g. "return your answer in this fomrat…") instructs it to return its response in the specified format.

Analyst Prompt

I am providing you with a list of events followed by their year. The purpose of this list is to analyze key events that were mentioned in the 1860 Hansard debates. Name some different events that might be relevant based on the list I provide you. Don't mention the same as provided by the list. Also provide a reason for why the events are relevant.
Return your answer in this format:
Event:
Reason:
These are the events from the list: (copy/paste contents of eventslist)

ChatGPT-4o Response

Event: Abolition of the Slave Trade (1807)
Reason: The abolition of the slave trade would be highly relevant in discussions about moral and economic reforms during the 1860 Hansard debates, particularly in the context of human rights and British colonial policies.
Event: Reform Act (1832)
Reason: This act, which expanded suffrage and restructured parliamentary representation, would likely influence discussions on electoral reform and representation in 1860.
Event: Crimean War (1853–1856)
Reason: The Crimean War had significant impacts on British foreign policy, military reforms, and public health improvements, all of which could be relevant topics of debate in 1860.
Event: Irish Potato Famine (1845–1852)
Reason: The famine's aftermath would remain a pressing issue, influencing debates about Irish governance, economic policies, and relief efforts.
Event: Great Exhibition (1851)
Reason: As a symbol of industrial progress and imperial power, the Great Exhibition could be discussed in terms of economic innovation and Britain's global role in the mid-19th century.
Event: Indian Rebellion (1857)
Reason: The rebellion was a turning point in British imperial governance, leading to significant debates about military strategy, colonial administration, and the justification of imperial rule.
Event: Public Health Act (1848)
Reason: This act, which addressed urban sanitation and public health, would be relevant to discussions about improving living conditions and public health policies in 1860.
Event: Establishment of the Penny Post (1840)
Reason: The reform in postal services, increasing communication efficiency and accessibility, could relate to debates on economic and infrastructural development.

Our initial prompt to the AI chatbot successfully guided it to identify and return events similar to the query. However, when we provided the chatbot with the entire contents of eventslist, the results lacked precision and specificity. This outcome suggests that more targeted guidance is necessary to focus on the specific characteristics of interest. To tailor the results to a specific research question, we will refine the prompt further by narrowing in on the events we are most interested in from `eventslist` and instructing the chatbot to return like-events, in this case, other names of meetings that occurred in the 1860s.

Analyst Prompt

I am providing you with a list of events followed by their year. The purpose of this list is to analyze key events that were mentioned in the 1860 Hansard debates. Name some different events that might be relevant based on the list I provide you. Don't mention the same as provided by the list. Also provide a reason for why the events are relevant based on the list I provide.
Return your answer like this:
Event:
Date:
Reason:
These are the events from the list: "greek war", "napoleonic wars."

ChatGPT-4o Response

Here are some different events that might be relevant based on your list:

Event: Chartist Movements
AI Assigned Date: 1838–1850
Reason: The Chartist movements, with their organized public gatherings for political reform, directly relate to the mentioned events like "public meeting" and "recent meeting," illustrating how assemblies have historically been used to drive societal change.

Event: Anti-Corn Law League Meetings
AI Assigned Date: 1839–1846
Reason: These meetings share a thematic connection to "public meeting" by demonstrating how public gatherings have been pivotal in shaping discourse around economic reform, similar to those discussed in the 1860 debates.

Event: Hyde Park Demonstrations
AI Assigned Date: 1855–1860
Reason: Hyde Park demonstrations resonate with "public meeting" and "recent meeting," showing how such gatherings functioned as platforms for political and social advocacy, reflecting the period's reliance on public discourse for progress.

Event: Meetings on the Reform Act of 1832
AI Assigned Date: 1831–1832
Reason: These meetings are akin to "yorkshire meeting" and "clontarf meeting," showcasing historical instances where regional or national public assemblies catalyzed significant legislative reforms, serving as a possible reference point in 1860.

Event: Fenian Meetings
AI Assigned Date: 1858–1860
Reason: While slightly later, Fenian meetings parallel "clontarf meeting" by highlighting Irish political concerns and illustrating how public gatherings were used to address issues of representation and governance.

Now that we have a more specific list of events that aligns with our interests, we can use them in our analysis. To use our new events, we will create a data frame for them. We can prompt the chatbot to return the needed code. Given that we are very familiar with creating data frames at this point, we can also proof the chatbot's results and ensure they match our desired output.

Analyst Prompt

Return this text as a data frame in R with two columns: a column for the event, named event_name and a column for the date, named AI Assigned Date. Write the events in lower case.

ChatGPT-4o Response

Here is the R code to create the requested data frame:

```
# Create the data frame

events_df <- data.frame(
  event_name = c(
    "crimean war",
    "russo-turkish war",
    "congress of vienna",
    "siege of acre",
    "battle of navarino",
    "schleswig-holstein wars"
  ),
  AI_Assigned_Date = c(
    "1853-1856",
    "1828-1829",
    "1814-1815",
    "1799",
    "1827",
    "1848-1852"
  )
)

# Print the data frame
print(events_data)
```

This will create a data frame in R with two columns: event_name (containing the event names in lowercase) and AI Assigned Date.

We can now test the chatbot's responses by running the code.

```r
# Create the data frame
events_df <- data.frame(
  event_name = c(
    "crimean war",
    "russo-turkish war",
    "congress of vienna",
    "siege of acre",
    "battle of navarino",
    "schleswig-holstein wars"
  ),
  AI_Assigned_Date = c(
    "1853-1856",
    "1828-1829",
    "1814-1815",
    "1799",
    "1827",
    "1848-1852"
  )
)

# Print the data frame
print(events_df)
```

```
##                 event_name AI_Assigned_Date
## 1              crimean war        1853-1856
## 2        russo-turkish war        1828-1829
## 3        congress of vienna       1814-1815
## 4           siege of acre              1799
## 5       battle of navarino             1827
## 6 schleswig-holstein wars        1848-1852
```

```
data("hansard_1860")
data("debate_metadata_1860")

hansard_1860 <- hansard_1860 %>%
  left_join(debate_metadata_1860)
```

```
## Joining with ‘by = join_by(sentence_id)‘
```

The following code filters the `hansard_1860` dataset to include rows where the `text` column matches any event name in `events_df$event_name`, using case-insensitive matching. It then adds a new column, `matched_event`, which contains the specific matched event name from the `text`. The result is a filtered data frame with an additional column identifying the matched event for each row.

```
# Match with each event and
# add a column to capture the matched event
new_matched_events <- hansard_1860 %>%
  filter(str_detect(
    text,
    regex(paste0("\\b(", paste(events_df$event_name, collapse = "|"), ")\\b"),
          ignore_case = TRUE))) %>%
  mutate(
    matched_event = str_extract(
      text,
      regex(paste0("\\b(", paste(events_df$event_name, collapse = "|"), ")\\b"),
            ignore_case = TRUE)))
```

The results show that not all of the events were mentioned word-for-word in the Hansard corpus; however, we successfully matched several, including the Crimean War, Congress of Vienna, and Battle of Navarino. While an AI chatbot can suggest a wider range of related terms to search for–and these terms might spark our curiosity–those terms may not actually appear in the corpus. For experts 19th-century history, it might

Table 8: Excerpt from Hansard 1850

sentence_id	text	speechdate	debate	matched_event
S3V0156P0_3555	As to the noble Earl's argument against the abandonment of a prohibitive duty on coal in the event of war, I have only to say that at the time of the Crimean war we did issue a proclamation prohibiting the export of contraband of war to countries north of Dantzic, but that we did not extend it to coal.	1860-02-20	QUESTION.	Crimean war
S3V0156P0_8154	The Crimean war, while it created an enormous extra demand for ships, curtailed in a very small degree the usual channels of commerce, and the consequence was that a larger number of ships was called into requisition than was necessary for the usual trade of the country.	1860-01-31	COMMITTEE MOVED FOR.	Crimean war
S3V0156P0_8400	Member for Sunderland himself, who, in a work which he had recently published, made mention of the adversity under which the shipping interest had laboured over since the date of the Crimean war, and declared the prediction that that adversity was only a mere passing cloud to have been completely falsified.	1860-01-31	COMMITTEE MOVED FOR.	Crimean war
S3V0156P0_10788	With regard to the claims of officers of the Land Transport Corps, a Committee of that House had sat, before which was brought the case of certain officers who had entered that corps under very peculiar circumstances, and who at the conclusion of the Crimean War were dismissed from that service.	1860-02-03	OBSERVATIONS.	Crimean War
S3V0156P0_14522	He did not think it possible to conceive waste like that which had taken place in disposing of the surplus stores after the Crimean war.	1860-02-10	COMMITTEE.	Crimean war

be expected that some phrases would not be found, but for others, this process can offer a useful way to explore which events were and were not mentioned in the data. This kind of exploration is part of an ongoing, iterative process—where analysts refine their searches by adding or adjusting keywords, either manually or with AI.

It is also important to note that the AI chatbot did not automatically fix formatting issues in the text, such as inconsistent capitalization. To obtain more reliable and visually meaningful results, the analyst may want to take extra processing steps to clean the data—such as making all text lowercase and fixing other formatting problems—before analysis.

```r
event_counts <- new_matched_events %>%
  count(matched_event, name = "count") %>%
  arrange(desc(count))

print(event_counts)
```

```
##        matched_event count
## 1        Crimean war   360
## 2        Crimean War   143
## 3 Congress of Vienna    42
## 4 battle of Navarino     5
```