Chapter 3: Investigating Speakers and Change Over Time Using Grouped Data

In previous chapters, we investigated how breaking up strings of text into individual words or multi-word phrases can give us insight into a corpus. We did things like count the number of words within an entire data frame. But historians often want to profile not merely the collective use of language, but also the role of specific speakers and how their language changed over time.

Understanding change over time begins with the question of what is to be understood over time. Is it a list of words referring to women? Is it an individual life? Do we care about the course of change over days, weeks, years, or decades – or within a single debate? For many historians, the answer might well be, "all of the above," which is fine. Counting one variable (say, references to women) by another variable of time (say, decades) is how we accomplish an analysis of change over time.

In this chapter, we will learn how to count over time – and the more general principle, which is analyzing one variable against another variable. We may count words by time, or words by speaker to find which speakers said these words the most.

In the language of the algorithm, understanding the association between variables is accomplished by "grouping" the data, allowing us to take other fields, like speaker name, year, and debate title, into account. This chapter will use these groupings to gain insight into the discourse of speakers in Parliament.

At this point, critical thinking about what we count is appropriate.The prospect of analyzing the individual writers in a corpus is an obvious way to understand the poles of debate. But when the subject of analysis is the parliament of Great Britain, one understandable reaction may be skepticism. Why study the language of individual speakers in Parliament when 19th-century British Parliament was mostly a small group of elite white men and not people who reflected the British population?

One answer is that parliamentary language is an index of social experience. We may come to the parliamentary debates interested in the experience of women or ethnicities; the research program here is neutral as to what the researcher wants to study.

Issues that rose to parliamentary attention were few. Ordinary people had to gather petitions to get parliament to entertain the vote for working people or women. When we look at parliament, we don't see all the labor that went into assembling the petition – but we do see a very important register of when discussions in parliament began to change. Using parliament as an index of political labor allows us to ask of any given subject, when did this subject merit institutional attention? Understanding the flow of power in parliament is a method useful to many researchers who are otherwise uninterested in identifying the characteristics of the "great men" of history.

In the exercises that follow, we anticipate that most researchers can see the value of finding out more about both the men who held powerful offices and the experience of women. We will therefore touch on both the mighty – including prime ministers and chancellors of the exchequer – and the vulnerable – showing how we can use data to navigate to the few women who testified in parliament. Careful navigation of data can teach us about both kinds of people.

Counting by Group to Find the Wordiest Speakers

Our first exercise is finding the most "wordy" Parliamentarians from 1830–that is, Parliamentarians who spoke the most words. Often, the Parliamentarians who spoke the most did so because fellow Parliamentarians trusted them to express the wants of their own social group.

To analyze speakers in Hansard, we will first need to import a new category of data from `hansardr`: `speaker_metadata`.

```
# load the hansardr library
library("hansardr")

# load the required data
data("speaker_metadata_1830")

head(speaker_metadata_1830)
```

```
##       sentence_id    speaker suggested_speaker ambiguous fuzzy_matched ignored
## 1 S3V0010P0_11508 Mr. Croker  john_croker_1539         0             0       0
## 2 S3V0010P0_11509 Mr. Croker  john_croker_1539         0             0       0
## 3 S3V0010P0_11510 Mr. Croker  john_croker_1539         0             0       0
## 4 S3V0010P0_11511 Mr. Croker  john_croker_1539         0             0       0
## 5 S3V0010P0_11512 Mr. Croker  john_croker_1539         0             0       0
## 6 S3V0010P0_11513 Mr. Croker  john_croker_1539         0             0       0
```

For now, we will just go over three relevant fields from `speaker_metadata`.

Each sentence from the Hansard corpus is assigned a unique ID, as represented by the `sentence_id` field. The speaker who stated a sentence is assigned the same ID. The shared key across datasets allows us to join the data from speaker_metadata to other data from `hansardr`, such as the debate text.

The other two important fields we will address are the `speaker` and the `suggested_speaker` fields.

The `speaker` field contains the speaker name as it was originally transcribed within the Hansard corpus. The resulting field contains inconsistencies and a lack of standardization that can make analysis of speakers difficult. A single speaker may have been recorded by different permutations of his first, middle and last name(s) (for example, "William Gladstone" may be transcribed as "William E. Gladstone", "W. Gladstone", or "Mr. Gladstone," to name just a few). In other cases, a speaker may have been called by a rotating office title (like "Prime Minister").

It is also the case that different speakers are recorded by the same name. For instance, two people named Sir Robert Peel served in Parliament during the 1820s and both are evoked by the same spelling of the name; the older Sir Robert Peel (1750-1830), the industrialist, briefly overlapped with his son, Sir Robert Peel (1841-46), the future prime minister. Meanwhile, "Mr. W. Gladstone" could refer to William Ewart Gladstone or his son William Henry Gladstone.

To make analysis of speaker names more challenging yet, during the digitization processes, optical character recognition (OCR) errors were introduced into the speaker names. Common OCR error include interpreting a lower case "L" as an "i," or the lower case letter "a" as an "o."

Because of these nuances, any naive attempt to count words by speaker would be guaranteed to produce wrong results. It was paramount, therefore, that in managing the `hansardr` data, we produce an authoritative index of actual speakers that corresponded to the facts. In data science, adding a column with information inking this authoritative index to the names as presented is called "metadata."

The curators of the dataset – a team of historians and data scientists run by the authors – stored the metadata about the speakers in the `suggested_speaker` field, building upon the work completed by the Digging into Linked Parliamentary Data Project and the Egger and Spirling database (Egger and Spirling, Winters and Steer).

The `suggested_speaker` field contains our suggestion for the true identity of the speaker. Importantly, the `suggested_speaker` field represents a marked improvement on existing data sources, tested by historians, but our data is not guaranteed for universal accuracy; hence it is only "suggested." In the suggested_speaker field, a unique id is given that refers to one unique speaker who was in parliament during the time covered by the database. While multiple speakers during the same period sometimes share the same name (like the two Gladstones or two Peels), each individual speaker is assigned a unique number. Thus, when we see the number 3104 in the suggested_speaker field, the curators of the database have made the inference that the W. Gladstone in question refers to William Gladstone, not his son William Henry.

In the following code, we will make our data set smaller and easier to work with by just selecting the "speaker," "suggested_speaker," and "text" fields before tokenizing the data into individual words.

```
library("tidytext")
library("tidyverse")
library("lubridate")

data("hansard_1830")
data("speaker_metadata_1830")

words_1830 <- hansard_1830 %>%
  left_join(speaker_metadata_1830) %>%
  select(speaker, suggested_speaker, text) %>%
  unnest_tokens(word, text)

head(words_1830)
```

```
##                 speaker suggested_speaker   word
## 1 The Duke of Buccleugh walter_scott_6566   rose
## 2 The Duke of Buccleugh walter_scott_6566     my
## 3 The Duke of Buccleugh walter_scott_6566  lords
## 4 The Duke of Buccleugh walter_scott_6566     in
## 5 The Duke of Buccleugh walter_scott_6566 rising
## 6 The Duke of Buccleugh walter_scott_6566     to
```

With a data frame containing fields for "speaker", "suggested_speaker", and "word," we are now in a position to perform a new kind of analysis to see the number of words each speaker contributed to Parliament using two new functions: `group_by()` and `summarize()`.

We use `group_by()` to organize the data by group before using summarize for our count. By using `group_by()` we can count the number of times one variable occurs, given the presence of another vari-

able. We can count the number of words spoken by a speaker. We could also use group_by to count the words spoken in a given year – or by a given speaker in a given year, and so on.

The arguments taken by the command `group_by()` tell the computer which variables to associate with each other. To group by speaker, we make the argument of `group_by()` speaker, like so:

```
words_per_speaker_1830 <- words_1830 %>%
  group_by(speaker)
```

Alternatively, we could group by suggested_speaker:

```
words_per_speaker_1830 <- words_1830 %>%
  group_by(suggested_speaker)
```

A useful function that often accompanies `group_by()` is `summarize()`, a command that tells the computer to apply a mathematical transformation to each of the groups created by group_by. Thus if we group_by(speaker) and then summarize(), we can count how many words each speaker spoke, or how many dates a speaker appeared on, or the first date when each speaker appeared.

The `summarize()` function can be used with any statistical transformation, for instance `mean()` or `max()`. Here, we will use `summarize()` with a function to count, which is `n()`. Finally, `arrange()` allows us to sort the speakers in descending order of total words spoken.

In the following code the argument `words_spoken = n()` tells `summarize()` to create a new column with the number of words that reflect each unique speaker

```
words_per_speaker_1830 <- words_1830 %>%
  group_by(speaker) %>%
  summarize(words_spoken = n()) %>%
  arrange(desc(words_spoken))

head(words_per_speaker_1830)
```

```
## # A tibble: 6 x 2
##   speaker                          words_spoken
##   <chr>                                   <int>
## 1 Mr. Hume                               856835
## 2 Mr. O'Connell                          823600
## 3 Sir Robert Peel                        822825
## 4 Lord John Russell                      782439
## 5 The Chancellor of the Exchequer        620830
## 6 Lord Brougham                          596096
```

Most students of British history will be familiar with the names of the speakers included in the `words_per_speaker_1830` variable, which include two prime ministers as well as one noted Irish orator and nationalist, Daniel O'Connell. The count above also shows us that among the most prolific speakers was the Chancellor of the Exchequer. How should we interpret this fact?

A naive analyst might assume that the Chancellor of the Exchequer was one person, the fifth most prolific speaker in the 1830s. Yet this assumption would be an error of interpretation. The number of words spoken by the Chancellor counts the speeches of several distinct individuals.

The careful analyst will be concerned about the speaker classified as "The Chancellor of the Exchequer," for many individuals held this post over the course of the years 1830-39, which we are looking at now. The Chancellor of the Exchequer is the head financial officer of the United Kingdom.

A more experienced researcher might begin by using the suggested_speaker field – not the speaker field – to count words spoken by each individual. Notice that the lines of code below differ from the lines of code above only in that the argument of the function `group_by()` has been changed from "speaker" to "suggested_speaker:"

```
words_per_speaker_1830 <- words_1830 %>%
  filter(suggested_speaker != "") %>%
  group_by(suggested_speaker) %>%
  summarize(words_spoken = n()) %>%
  arrange(desc(words_spoken))

head(words_per_speaker_1830)
```

```
## # A tibble: 6 x 2
##   suggested_speaker    words_spoken
##   <chr>                       <int>
## 1 robert_peel_1664           1224576
## 2 henry_brougham_1679        1180672
## 3 joseph_hume_1712            875951
## 4 daniel_oconnell_2552        808577
## 5 john_russell_1885           802418
## 6 thomas_rice_2286            647551
```

In this view, we see the first and last names of individual speakers as well as their ID number from parliament (that is, Robert Peel is id 1664; 1664 is not, in this case, a date).

Nevertheless, our first pass revealed something interesting – as byways in the archives often do. It is interesting that Hansard represents the Chancellor of the Exchequer speaking more words than any other office – including that of Prime Minister – at least for the 1830s. The Chancellor of the Exchequer oversees the work of the Treasury. In many cases, the Chancellor of the Exchequer was a preliminary step to becoming Prime Minister. Why was the Chancellor of the Exchequer so important? And what were the various Chancellors talking about that required so much speechifying?

A Research Question: What were the Chancellors of the Exchequer talking about?

It may seem arbitrary to investigate the Chancellor of the Exchequer, but the office provides a perfect opportunity to test how the metadata indexing individual speakers works. Next, will use the suggested_speaker field to pull apart the many individuals associated with the office.

Because the dataset includes metadata about each speaker and how they are described, we can find the many individuals who occupied the office of the Chancellor of the Exchequer, and we can do research on them instead. In a sense, it's a happy accident of our research process that we used `group_by()` applied to the speaker field rather than the authoritative suggested_speaker field, because the speaker field tells us about the way that individuals were actually designated in Hansard, including their office name.

Before we proceed, we should try to refine our research question to provide maximum historical insight so that we can start to imagine how our quantitative research process becomes worthwhile. For instance, we can ask questions about the diversity of views of the individuals who and we can ask questions about the office.

We would expect most of the Chancellor of the Exchequer's speeches in parliament to reflect a predominant concern with taxation and spending. However, it is also possible that different individuals who held this post prioritized different issues. How different were they, the Chancellors of the Exchequer for 1830-39?

To analyze the different speakers, the code that follows will use `group_by()` to group the speeches made by Chancellors of the Exchequer by suggested_speaker – the actual individuals behind the office. Then, we can inspect the number of words used by each individual Chancellor, comparing their favorite words.

A Custom Stop Words List

Before we proceed, a first step is to "clean" our data. In previous chapters we used an existing stop words list from `tidytext`. The `hansardr` library also provides its own stop words list.

It may be the case, however, that an analyst wants to curate their own stop words list that caters to their specific data set or research question. In the following code we create a custom stop words list by assigning a list of words to a tibble (a `tidyverse`-style data frame). For our analysis, suppose we want to restrict all words referring to parliament, ideas, or procedures; we only want to look at words that are more or less substantive about the subjects being discussed in parliament.

An appropriate stopwords list might look like this:

```
custom_stop_words = tibble(word = c("hon", "speaker", "house", "question", "lord", "bill",
"committee", "duty", "country", "time", "amount", "government", "proposed", "law",
"measure", "learned", "law", "sir", "respect", "public", "gentleman", "gentlemen",
"friend", "noble", "parliament", "expenditure", "revenue", "tax", "principle", "proposal",
"consideration", "duties", "stated", "parliament", "act", "opinion", "inquiry", "effect",
"subject", "object", "motion", "baronet", "crown", "propose", "estimates", "sum",
"account", "ministers", "majesty", "proposition", "persons", "principles", "service",
"found", "propositions", "office", "matter", "statement", "paid", "increase", "moved",
"means", "considerable", "supply", "intention", "debt", "received", "expense", "estimate",
"charges", "resolution", "notice", "report", "parties", "party", "surplus", "commons",
"parties", "class", "commission", "form", "answer", "commissioners", "appointment",
"officers", "saving", "appeared", "granted", "late", "day", "future", "opportunity",
"saving", "officers", "information", "extent", "authority", "session", "justice",
"believed", "support", "character", "carried", "plan", "paper", "clause",  "opposite",
"system", "argument", "rate", "considered", "bills", "increase", "result", "reference",
"prepared", "laid", "portion", "passed", "intended", "chancellor", "taxation", "classes",
"appointed", "established", "parish", "confidence", "evidence", "pensions", "bring",
"income", "connected", "referred", "objection", "hoped", "adopted", "chancellor",
```

```
"account", "ministers", "occasion", "reduction", "reductions", "service", "services",
"attention", "vote", "brought", "forward", "purpose", "wish", "called", "period",
"money", "purpose", "power", "charge", "view", "circumstances", "trade", "taxes", "list",
"civil", "wished", "people", "discussion", "applied", "proper", "taking", "amendment",
"statements", "repeal", "papers", "fund", "feeling", "measures", "minister", "purposes",
"payment", "stock", "increased", "bound", "grant", "grounds", "doubt", "applied", "stamp",
"manner", "conduct", "exchequer", "provide", "reduced", "table", "difficulties",
"anxious", "compared", "provide", "king", "individual", "expressed", "hear", "hope",
"establishment", "debate", "contrary", "instance", "introduced", "call", "lords",
"existing", "half", "head", "offices", "views", "leave", "respecting", "speech", "feel",
"excise", "effected", "economy", "night", "price", "majority", "reason", "disposed",
"held", "claims", "leave", "required", "moment", "admitted", "ready", "words", "issue",
"allowed", "true", "treasury", "administration", "consent", "advantage", "business",
"calculated", "enter", "fair","nature", "oppose", "reign", "honour", "mode", "reduce",
"difficulty", "favour", "credit", "adopt", "additional", "individuals", "mind", "sale",
"times", "amounted", "deficiency", "laws", "til", "agreed", "alluded", "mentioned",
"meant", "heard", "feelings", "engaged", "consolidated", "conclusion", "circulation",
"begged", "calculations", "afford", "acts", "acted", "giving", "pledge", "matters",
"contract", "proceed", "determined", "satisfaction", "declare", "giving", "spirits",
"resolutions", "pursue", "opposition", "force","pursue", "force", "opinions", "refer",
"produce", "appeal", "pay", "terms", "private", "reform", "observations", "entitles",
"laws", "alteration", "due", "gallant", "joint", "undoubtedly", "alluded", "benefit",
"consequence", "perfectly", "trusted", "loan", "proceed", "cent", "regard", "political",
"royal", "carry", "importance", "proceedings", "aware", "rates", "similar", "examination",
"details", "told", "supposed", "sufficient", "person", "necessity", "millions",
"emoluments", "difference", "department", "till", "submit", "lay", "imposed", "claim",
"branches", "recommended", "privy", "personal", "select", "salary", "revenues",
"preceding", "practice", "move", "meet", "hand", "funds", "formed", "expenses",
"establishments", "entitled", "dimunition", "settlement", "sense", "arrangement",
"address", "sufficient", "local", "clergy", "relief", "necessity", "application", "claim",
"responsibility", "situation", "satisfactory", "ground", "revenues", "quarter", "change",
"founded", "secretary", "provided", "read", "commutation", "body", "provision",
"difference", "composition", "condition", "hands", "provision", "capital",
"responsibility", "press", "knowledge", "raised", "friends", "altogether", "meet", "hand",
"free", "difference", "claim", "necessity", "arguments", "left", "single", "founded",
"arrangement", "admit", "total", "told", "operation", "apply", "shown", "person",
"decision", "actual", "returns", "return", "restrictions", "national", "limited",
"hundred", "existed", "council", "charged", "annual", "allowances", "proceeding", "word",
"existed", "fairly", "loss", "spirit", "west", "sum"))
```

We are ready to filter and clean our data and explore the langauge of speakers called Chancellor of the Exchequer.

Counting the Words of Each Speaker

In the following code we first remove instances of John Spencer and Rigby Wason, who spoke so few words as Chancellor of the Exchequer that he contributes little to our project. Then we eliminate numbers, stop

words, and custom stop words before counting each speaker's words with `group_by()` and `summarize()`.

```
chancellor_of_the_exchequeur <- words_1830 %>%
  filter(str_detect(speaker, "Exchequer"),
         str_detect(word, "[a-z]"),
         suggested_speaker != "john_spencer_1234",
         suggested_speaker != "rigby_wason_3024",
         suggested_speaker != "",
         suggested_speaker != "chancellor of the exchequerr") %>%
   anti_join(stop_words) %>%
   anti_join(custom_stop_words) %>%
   group_by(suggested_speaker, word) %>%
   summarize(n = n()) %>%
   top_n(35) %>%
   ungroup()

chancellor_of_the_exchequeur %>%
  sample_n(5) %>%
  head()
```

```
## # A tibble: 5 x 3
##   suggested_speaker   word            n
##   <chr>               <chr>       <int>
## 1 henry_goulburn_1824 petitioners    15
## 2 thomas_rice_2286    pension       149
## 3 henry_goulburn_1824 distress       24
## 4 henry_goulburn_1824 branch         15
## 5 thomas_rice_2286    property      256
```

Note that in the table above, the "word" is annotated with information about the speaker. Annotation is used to prevent confusion about what is being counted; we're not just counting how many times a word appears, but how many times each speaker said it.

Next, let's visualize the data so that we can see how many words are spoken.

```
chancellor_of_the_exchequeur <- chancellor_of_the_exchequeur %>%
  mutate(word = reorder_within(word, n, suggested_speaker))

ggplot(data = chancellor_of_the_exchequeur,
       aes(x = word, y = n)) +
  geom_col() +
  scale_x_reordered() +
  coord_flip() +
  facet_wrap(~suggested_speaker, scales = "free") +
  labs(x = "word", y = "speaker") +
  ggtitle("Favorite Words of Each Chancellor of the Exchequer in the 1830s")
```

The results show that the individuals who occupied the post of Chancellor of the Exchequer in fact represented a diversity of interests, ranging from Goulburn's interest in colonial commodities like "tobacco" to

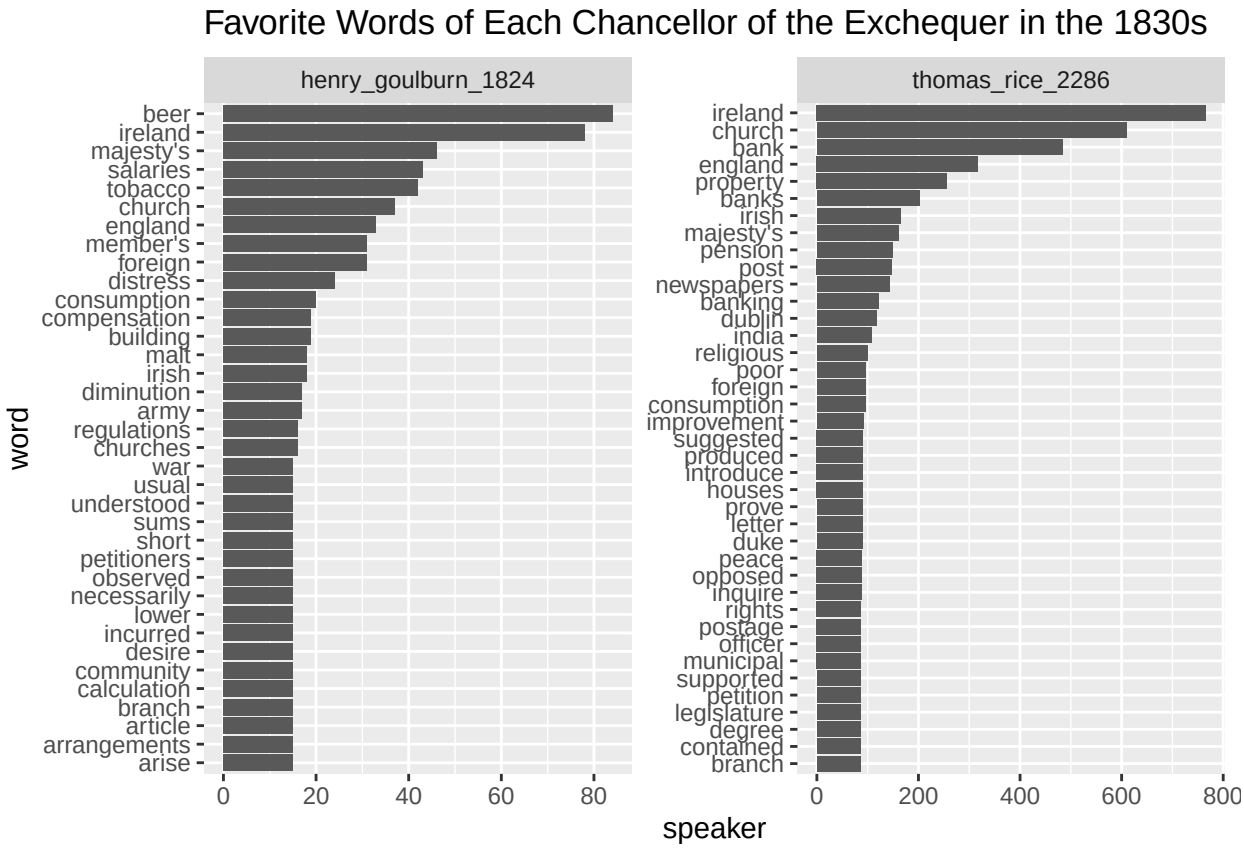


Figure 1: Most frequently used words by each Chancellor of the Exchequer in the 1830s. Bars show the top repeated terms spoken by Henry Goulburn (left panel) and Thomas Spring Rice (right panel). Counts on the x-axis represent the number of times each word appeared in that Chancellor's speeches during the decade.

Rice's interest in the tension between the property and banking interests. Their territorial interests varied as wel Goulbourn emphasized England and Ireland, while Rice emphasized Ireland and India

Goulburn is interested in talking about commodities such as beer, tobacco, and malt; Peel is interested in addressing questions about how taxes relate to the church and its tithes, as well as political issues about how dissenters, Protestants and Catholics. Peel also thinks about commodities, but his commodities are different than Goulburn's; they include malt, land, and barley. Rice takes the issues in a different direction, reflecting on Dublin, India, banks, newspapers, and pensions. He shares with Peel a concern about property or land taxes.

Interpretation always represents an intellectual challenge. A list of most frequent words is not necessarily an index of importance: the most frequently-mentioned terms are typically the terms most subject to debate. Robert Peel, for instance, does not name Ireland's church tithes because he was interested in protecting the church in Ireland; to the contrary, he wished to abolish the system of tithes. Nor does the word list give us material for interpreting the rest of Peel's politics; the word list does nothing to tell the uninformed analyst about the movement for the abolition of taxes on grain, which forms a background to interpreting Peel's mentions of barley. The word list tells us little about the presence in 1830s Ireland of two churches, Roman Catholic and the Church of England. Without further reading, we might never known that Goulburn was famous for arguing for reductions on taxation (although with this knowledge, we might become interested in the word "lower"). The analyst who comes to the Hansard corpus without a background in British history would gain greater insight by reading secondary sources before interpreting the graph.

After some basic background research, however, the word lists can help produce insight, cluing the reader into differences such as Goulburn's relative willingness to talk about "distress," Peel's interest in "education," or Rice's interest in the post office. With word lists, we retrieve important clues about the salient differences contributed to political debates by individual ministers. A researcher interested in the careers of Chancellors of the Exchequer might start here before reading more deeply.

There is no rule of thumb for how many words to look at, but more words is always better. Distant reading, by its nature, is a shortcut to making sense of long passages of text. We want to take our time with these words, not rush to conclusions, and not merely read the top five. If we expanded the list of words, we might get more insight – especially if we have gone to the trouble of reading more about the work of Goulborn and Rice during the 1830s and we have specific questions about their priorities. The researcher is welcome to adjust upwards the number of words shown in these charts by raising the number in the code `top_n(35)` – although at a certain point, legibility will require new code to expand the bounding box that limits how big a visualization is.

How new is the information we have gleaned to readers of British history? We have long known that issues of what was taxed and how are a basic matter for politics of class, empire and nation; everyone wants someone else to pay for the government. The research modeled in this textbook has a pedagogical function, and for that reason, we often present in these chapters examples that could be improved or investigated further, and the above chart is one of those.

Critical Thinking About Our Results

In any research process, it is important to reflect on how the choices we have made impact our analysis, and how making different choices might have produced different results. For some researchers, the list in `custom_stop_words` may unnecessarily eliminate words whose usage they might want to track. Some researchers will want to identify words for rhetorical statements such as "hoped" or "feeling"; others may be intrigued about how different speakers refer to "information" or "knowledge." In general, I have opted for

a deep list which includes every general allusion to governance, institutions, discovery, and communication. The beauty of custom lists is that each researcher can easily adjust the words to match their interests Analysts should understand how carefully they must tailor a custom stop words list to their project, and how much of a difference additions and subtractions make. Using a custom stop word list is very much an artisinal skill of research; it involves much the same skills of judgment, awareness, curiosity, and sensitivity to background issues from theory and secondary sources as does careful reading of primary sources.

One complication of stopwords lists is that to produce functional results, a custom stop words list must often be long; the hundreds of performative, rhetorical, and governmental words listed below is not a perfect list, and it would not work for every exercise. In later chapters, we will investigate approaches such as differential measurement which allow the reader to skip over custom stop words lists to obtain information about what distinguishes one speaker from another or one year from the next. No process is perfect, however, and analysts must often resort to some kind of custom filter to get closer to information that is useful for their project. There is no silver bullet for creating an absolutely accurate tool where text-mining research always produces useful information; rather, iterative inquiry, paired with background reading and in-depth reading of primary sources, is the basis upon which all insight is ultimately made.

Working With Dates

Knowing who spoke the most in parliament is useful. But very often insight comes not from examining how counts change over time. What if we want to know not merely who spoke the most, but who was the top speaker for each year in parliament? Our `hansard_1830` data frame lists a date in the "speechdate" column, formatted as a four-digit year, two-digit month, and two digit day. To easily track speeches by year, we can add a new "field" – that is, a column – listing just the year of each speech.

Very frequently, when working with information about dates, we need to extract the month, day, or year from a data set. The `lubridate` package makes it easy to extract this information from dates in any format.

```
date1 <- mdy("6/24/1819")
date1
```

```
## [1] "1819-06-24"
```

```
year1 <- year(date1)
year1
```

```
## [1] 1819
```

We will use the `year()` function from `lubridate` with `mutate()` to create a new column that has just the extracted year from the `speechdate` field.

```
library("kableExtra")

data("debate_metadata_1830")

hansard_1830_w_year <- hansard_1830 %>%
```

```
  left_join(debate_metadata_1830) %>%
  mutate(year = year(speechdate))

# Create a table preview of the dataset
# - Select the first rows (head())
# - Shorten long text for readability (str_trunc())
# - Format the table for PDF output (kable(), kable_styling())
# - Adjust column widths for better layout (column_spec())
hansard_1830_w_year %>%
  head() %>%
  mutate(text = str_trunc(text, 120)) %>%
  kable("latex", booktabs = TRUE) %>%
  kable_styling(latex_options = c("hold_position"),
                full_width = FALSE,
                position = "left") %>%
  column_spec(2, width = "6cm") %>%
  column_spec(4, width = "4cm")
```

sentence_id	text	speechdate	debate	year
S2V0022P0_0	rose:— My Lords;—In rising to move that a humble Address be presented to his Majesty, in answer to the most gracious ...	1830-02-04	ADDRESS ON THE LORDS COMMISSIONERS SPEECH.]	1830
S2V0022P0_1	It would be presumptuous in one so young and inexperienced as I am, and who have had the honour of a seat in your Hou...	1830-02-04	ADDRESS ON THE LORDS COMMISSIONERS SPEECH.]	1830
S2V0022P0_2	I shall, therefore, confine myself to such few observations and reasons as may occur to me, claiming, at the same tim...	1830-02-04	ADDRESS ON THE LORDS COMMISSIONERS SPEECH.]	1830
S2V0022P0_3	My lords, the strong assurances which his Majesty has been pleased to inform us he still continues to receive from th...	1830-02-04	ADDRESS ON THE LORDS COMMISSIONERS SPEECH.]	1830
S2V0022P0_4	This nation is too much involved in the general interests of Europe not to view with satisfaction the intelligence th...	1830-02-04	ADDRESS ON THE LORDS COMMISSIONERS SPEECH.]	1830
S2V0022P0_5	We cannot but hear, my lords, with satisfaction, of his Majesty's unremitting offices with his allies to carry into e...	1830-02-04	ADDRESS ON THE LORDS COMMISSIONERS SPEECH.]	1830

Notice that the resulting dataset has a new field, "year."

At this point in the chapter, we already created `words_1830`, a tokenized dataset with one row per word.

That dataframe is very large, and we will not use it again. If we keep it while also creating a second tokenized dataset (this time with a year column), we may run into memory limits. So to continue our analysis, we will do two things: - Remove the old tokenized dataset (`words_1830`) to free memory. - Tokenize the speeches again, but now we keep the year column so we can count words by year (and by speaker within each year).

```
rm(words_1830)
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  4549492 243.0   10564195  564.2   9200965  491.4
## Vcells 43607478 332.7  268102173 2045.5 333693092 2545.9
```

```
words_1830_w_year <- hansard_1830_w_year %>%
  left_join(speaker_metadata_1830) %>%
  select(speaker, suggested_speaker, text, year) %>%
  unnest_tokens(word, text)

words_1830_w_year %>%
  head() %>%
  select(speaker, word, year) %>%
  kable() #look at the first few columns of the dataset
```

speaker	word	year
The Duke of Buccleugh	rose	1830
The Duke of Buccleugh	my	1830
The Duke of Buccleugh	lords	1830
The Duke of Buccleugh	in	1830
The Duke of Buccleugh	rising	1830
The Duke of Buccleugh	to	1830

Now that we have a field for the year of each speech, we can return to "grouping" data to execute a faceted count where we count words by speaker and year. When we used the command `group_by()` above, we used it with one argument. But `group_by()` can take multiple arguments, allowing the analyst to perform grouped counts that involve multiple dimensions of the dataset. Next, we will use `group_by()` on two data fields at the same time – the "speaker" and "year" columns of as the arguments of `group_by()`, as in `group_by(speaker, year)`.

```
words_per_speaker_1830 <- words_1830_w_year %>%
  filter(suggested_speaker != "") %>%
  group_by(speaker, year) %>%
  summarize(speaker_words_per_year = n()) %>%
  arrange(desc(speaker_words_per_year))

head(words_per_speaker_1830)
```

```
## # A tibble: 6 x 3
## # Groups:   speaker [4]
##   speaker              year speaker_words_per_year
##   <chr>               <dbl>                  <int>
## 1 Lord Brougham        1839                 200436
## 2 Lord Althorp         1831                 180573
## 3 Lord Brougham        1838                 175070
## 4 The Lord Chancellor  1831                 161729
## 5 Mr. O'Connell        1834                 158617
## 6 Lord Althorp         1833                 154948
```

Notice that the result of grouping and counting is a data set with three columns: "speaker," "year," and "speaker_words_per_year." The last column is the count for each speaker per year – a field that makes it possible to analyze change over time.

When we use the `group_by()` command, the fields we use for grouping are preserved in the output. Two columns are the names of the facets we used for grouping – "speaker" and "year" – while the last column, "words_spoken," is the count of how many words are recorded for each speaker per year.

Women in 19th-century Parliament

For the most part, an analysis of speakers in Parliament will take the researcher down the road of the many elite white men who assumed powerful positions or passed honorary titles to their sons. Only 12 named women spoke in Parliament throughout the 19th-century. One woman who spoke the most was Mrs. Walrand from the 1824 debate, "Motion Respecting the Trial and Condemnation of Missionary Smith at Demerara." Exploring her role in this debate can give us insight and speculation into the criteria that had to be set in place for a woman to appear as a speaker and instrument to the power of Parliament.

"Motion Respecting the Trial and Condemnation of Missionary Smith at Demerara" reviews the trail, conviction, and death of John Smith, an Methodist missionary from England assigned to the British slave colony of Demerara in South America. Mr. Smith's trial accused him of assisting a slave rebellion that took place in Demerara in August of 1823. MP Brougham brought the topic of Mr. Smith's trial before Parliament to argue that the legal procedings convicting Mr. Smith were unjust and strayed from the guidance of Britain to the leaders of its Demerara colony. One could read his testimony as an early foundation for Britian's movement towards the Slavery Abolition Act of 1833, but his concerns–as well as the testimonies of other speakers–are nonetheless intertwined within a biased legal institution.

Mrs. Walrand spoke as a witness to the slave rebellion and served as a character witness condemning Mr. Smith. According to her testimony, she saw first hand the brutal violence of the rebells, having been a "defenceless lady" fired at by these "savages."

```
data("hansard_1820")
data("speaker_metadata_1820")
data("file_metadata_1820")

mrs_walrand_speaker_metadata <- speaker_metadata_1820 %>%
  filter(str_detect(speaker, regex("Mrs(.*)Walrand", ignore_case = T)))
```

Table 3.2: Sentences uttered by Mrs. Walrand in the 1820 Hansard debates.

debate_id	text
5142	proceeds: ""
5142	He (Mr. Forbes) said, how he envied Mr. Tucker his immediate death; and seemed in the most excruciating agony, but pe...
5142	I entreated the guard, in the name of every principle of humanity, just to let me send to Golden Grove, the next esta...
5142	Each of the guards, at different times, Murphy, Rodney, and others, refused.
5142	The man died at half past twelve that night.
5142	In the course of the forenoon of Tuesday, Murphy (the man since executed) came into the gallery of the sick-house, an...

```
mrs_walrand_sentences <- left_join(mrs_walrand_speaker_metadata,
                                   file_metadata_1820, by = "sentence_id") %>%
  left_join(., hansard_1820, by = "sentence_id") %>%
  select(debate_id, text)

mrs_walrand_sentences %>%
  head() %>%
  mutate(text = str_trunc(text, 120)) %>%
  kable(format = "latex",
        booktabs = TRUE,
        caption = "Sentences uttered by Mrs. Walrand in the 1820 Hansard debates.") %>%
  kable_styling(full_width = FALSE, position = "left") %>%
  column_spec(2, width = "10cm")
```

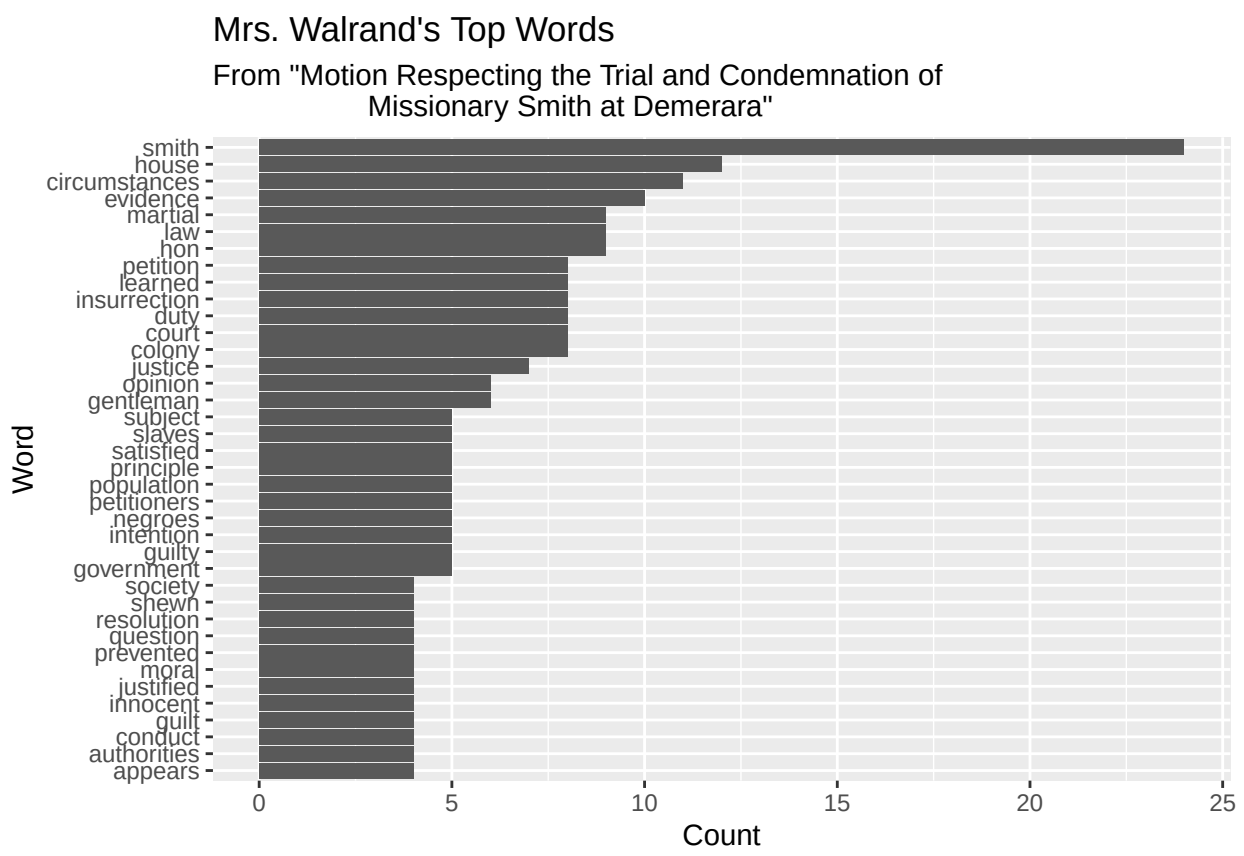
```
entire_debate <- hansard_1820 %>%
  left_join(file_metadata_1820, hansard_1820, by = "sentence_id") %>%
  filter(debate_id == 5142) %>%
  select("text")

debate_table <- tibble(text = head(entire_debate$text, 20))

debate_table %>%
  kable(format = "latex",
        booktabs = TRUE,
        caption = "Excerpt of Debate Text (First 20 Sentences)") %>%
  kable_styling(latex_options = c("hold_position", "scale_down"),
                full_width = FALSE,
                position = "left") %>%
  column_spec(1, width = "16cm")
```

```
mrs_walrand_top_words <- left_join(mrs_walrand_speaker_metadata, hansard_1820) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  filter(is.na(as.numeric(word))) %>%
  group_by(speaker, word) %>%
  summarize(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(30)

ggplot(data = mrs_walrand_top_words,
       aes(x = reorder(word, n), y = n)) +
  geom_col() +
  coord_flip() +
  labs(title = "Mrs. Walrand\'s Top Words",
       subtitle = "From \"Motion Respecting the Trial and Condemnation of
                   Missionary Smith at Demerara\"",
       x = "Word",
       y = "Count")
```



Investigating an Event

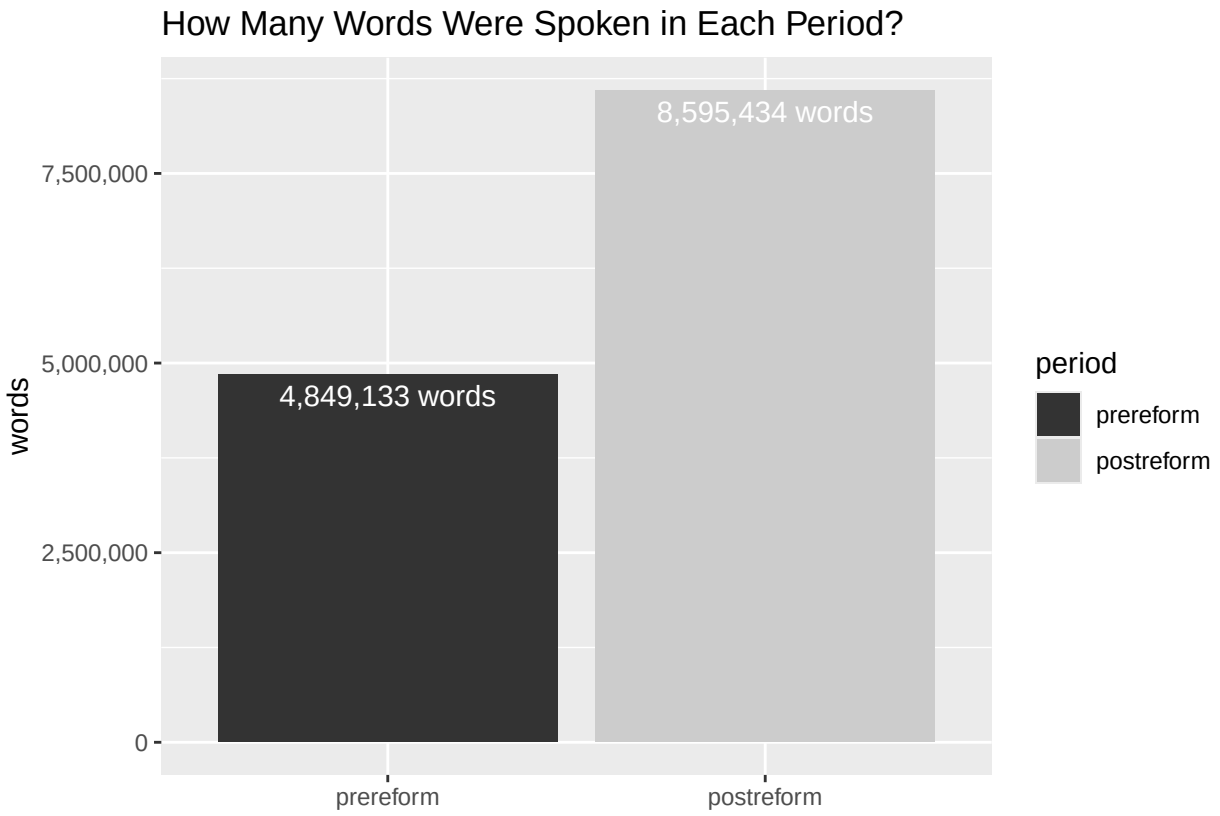
Next, let's move towards investigating an event.

First, let's organize the data into two groupings – the words spoken prereform and those spoken postreform.

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

Let's break the data up into individual words and count how many words are in each category.

```
## Joining with `by = join_by(word)`
## Joining with `by = join_by(word)`
## Attaching package: 'scales'
## The following object is masked from 'package:purrr':
##
## discard
## The following object is masked from 'package:readr':
##
## col_factor
```



Many more words were spoken in the ten years after the reform than the ten years before. How many more? Text mining means we never have to guess.

```
## [1] 3746301
```

Next, let's look at the actual words spoken in each period and see how they compare.

The analyst will discern that there are only very minor differences between the top words of both categories. What are we to make of that?

We should begin by admitting that these words tell us little that's new about parliament or history. The word "house" was frequently used in parliament to refer to the House of Commons and House of Lords, the two chambers of parliament. Members of the houses traditionally addressed each other as "honorable sir" (abbreviated "hon" in Hansard) and (when addressing peers of the realm) "noble sir," "noble sirs," or "my lord." The general matters of parliament concerned the state of the "country," the passing of "bills" into law, the activities of the "goverment," various other "subjects," and the official matter of "questions" – a formal category of address about the business of parliament. To find the top words offers no real insight to any analyst who understands the basic business of parliament.

The only apparent difference in this table occurs in the words "time" – a top-ten word prereform but not postreform – and "Ireland" – a top-ten word postreform and prereform. If we looked at more than the top-ten words, even these differences would likely appear slight; the two words missing in this table are almost

surely in the top twenty list for both periods – something we can easily check by adjusting the command *slice_head(n=10)* from 10 to 20.

In short, this step provides us with what data scientists refer to as a "validation" – that is, we have confirmed that the method of counting words tells us that the speech of parliament concerns the business of parliament.

This kind of validation step is crucial to knowing that we are on the right track. If our attempt at counting (or any other statistical analysis) suggested that parliamentary speech in general concerned squids, whales, and cephalopods, or even obscure names of towns in Ireland, we would guess that something had gone wrong. Throughout the process of data analysis, we should use basic statistical measures as validation that the general idea is working. If our analysis tells us anything that doesn't make sense, we should rush to check our method – not revisit the laws of history.

Strategies for Reaching Insight

How do we move from validation to insight? There are many strategies. Let's think through several:

- 1) We can read a longer list of top words for each period, for instance the top 500 instead of the top 10. Keeping track of minute changes may tell us something. On the other hand, there will be a great many words to look at. If the differences are slight (like a #9 vs a #12 position for a certain word from one period to the next) we risk losing track of which words are important for which periods.
- 2) We can look at other aspects of the dataset, for instance the speaker column, asking how many words were spoken by each speaker in the prereform and postreform periods.
- 3) We can use a variety of statistical methods to aggregate information about the speakers that were the most different and the words whose expression was most different in the prereform and postreform periods.

In the sections that follow, we will undertake each of these steps in turn. They will give us different answers.

Step #1 is easily accomplished by a quick adjustment in code, tweaking the command *slice_head(n=10)* from 10 to 500. Please change the command for youself and inspect the results. Is it easy to interpret them?

Step #2 is easily accomplished by revisiting the code in greater depth. We won't need any new commands, but we will need to apply the functions we've already seen – group_by, summarize, and n() – to the "speaker" field rather than the "year" field.

Let's do that now to answer the question, how many words did each speaker speak in the prereform and postreform period?

Investigating Speakers' Wordcounts, Pre- and Post-Reform

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

```
## # A tibble: 6 x 4
```

```
##   suggested_speaker        year wordcount period
##   <chr>                   <dbl>     <int> <chr>
## 1 abraham_hume_1561        1826       702 prereform
## 2 adolphus_dalrymple_1968  1831       457 prereform
## 3 adolphus_dalrymple_1968  1832      1699 prereform
## 4 alexander_baring_1475    1830      5671 prereform
## 5 alexander_baring_1475    1831      7135 prereform
## 6 alexander_dawson_2455    1826      1935 prereform
```

Questions About Data Quality

***STEPH – some notes about speaker data quality would be good here. Lots of variation in quality – including "A Member" who only spoke 8 words. Maybe give a history of how we cleaned the data, how good it is, and the mechanisms available for improving the data. We could also use this opportunity to help users to analyze what's strong, what's less strong about the data. Looks like there are some irregularities that the team could have found if they'd just deleted punctuation (e.g. "A Member" and "A Member. —"; presumably "The Earl of Carnarvon").

Because the team concentrated on speakers who sopke frequently, the data is much better if we look at the speakers who contributed more than a few sentences.

```
## # A tibble: 5 x 2
##   suggested_speaker    wordsperspeaker
##   <chr>                          <int>
## 1 robert_peel_1664             1982866
## 2 henry_brougham_1679          1616327
## 3 joseph_hume_1712             1336979
## 4 john_russell_1885            1046022
## 5 daniel_oconnell_2552          811831
```

Here we can see one potential problem – Lord Henry Brougham appears both as "Lord Brougham" and as "henry_brougham_1679" when in reality these are both the same person. The majority of speakers look much better, though.

Investigating Speaker Wordcount With Joins

With the data frame both_periods_speakers, we have wordcounts for each speaker for pre- and postreform. There's a great deal we can do with this data. First, let's perform some basic counts to help us to understand, in raw numbers, the basic features of prereform and postreform parliament.

We'll start by asking how many members of parliament stood for the ten years pre and post reform. We will label each with a "class" called "prereform only," "postreform only," or "continuous speakers." Note that these "classes" are fundamentally different than the "period" categorization we applied above, where we labeled speeches given before the 1832 reform law as "prereform" and those after the law as "postreform." Instead of labeling the years, we want to identify the speakers that spoke only in one period or the other, and those who spoke in both.

To identify these speakers, we will begin by returning to an earlier data frame – "prereform_speakers_wordcount" – and using a process called a "join" to exclude those speakers who also appear in the dataset "postreform_speakers_wordcount."

Information scientists theorize several ways to combine data through what is called "set theory." Two of the simplest joins are known as the "union" of two sets – what they have in common – and the "difference" between two sets – what is in one but not the other.

To find the prereform-only and postreform-only speakers, we will find the "difference" between the two sets by using a function named "anti_join()." The anti_join() function excludes from one dataset all the data that appear in a second dataset. If we begin with a dataset of all members of parliament who spoke prereform and perform an "anti-join" with the set of speakers who spoke postreform, the resulting dataset is a list of speakers who *only* spoke prereform.

```
## # A tibble: 6 x 2
##   suggested_speaker     class
##   <chr>                 <chr>
## 1 abraham_hume_1561     prereform only
## 2 alexander_baring_1475 prereform only
## 3 alexander_baring_1475 prereform only
## 4 alexander_dawson_2455 prereform only
## 5 alexander_dawson_2455 prereform only
## 6 alexander_dawson_2455 prereform only
```

How many speakers are there in our dataset? To answer the question, we just need to know the number of rows, given that the dataset lists one distinct speaker for each row. The function *nrow()* will provide an answer to how many rows there are in any dataset.

```
## [1] 1484
```

Performing the same operation again, beginning with postreform speeches and removing the rows of data that appear prereform, yields a dataset of speakers who *only* spoke after the reform.

```
## Joining with `by = join_by(suggested_speaker)`
```

```
## # A tibble: 6 x 2
##   suggested_speaker  class
##   <chr>              <chr>
## 1 aaron_chapman_3235 postreform only
## 2 aaron_chapman_3235 postreform only
## 3 aaron_chapman_3235 postreform only
## 4 aaron_chapman_3235 postreform only
## 5 aaron_chapman_3235 postreform only
## 6 aaron_chapman_3235 postreform only
```

How many speakers were there postreform?

```
## [1] 2438
```

We now have two sets of numbers – the count of unique speakers who spoke 1822-32 and 1832-42. These numbers are new to history; there has never before been a count of these speakers. We can learn something from them. Before text mining, it was impossible to get a real impression of how many speakers were active in parliament before and after 1832. But the simple ability to count allows us to add detail to our accounts of the reform and its mechanisms.

The basic finding is that there were more speakers after the reform period. This makes sense, because the reforms of 1832 introduced more popular consent into the democratic process. No longer were members appointed in "pocket buroughs" by aristocrats who held the sole authority for electing the representative. Now, all seats were open to a popular vote (albeit of those rich enough to pay at least £10 per year in taxes – no mean sum). If we know that the number of seats in the House of Commons were set at 658 before and after the reform (which we learn by consulting a secondary source, XXXXXX), the numbers tell us that this reform introduced more variation and turnover in who occupied the seats of parliament.

Substantively more different members of parliament were elected in the ten years following 1832 than in the decade before. *How many more members were there among the postreform only members than the prereform only members?*

```
## [1] 954
```

We can also investigate speakers from a third category – those who were continuously in parliament, who held office before the reform and who were able to secure election after the reform as well. How many such speakers were there?

Please note that we do not have the data to ask which members of parliament who held office before 1832 were *elected* to office after 1832. We do not have the data to support that question. What we have in Hansard is only a record of who was recorded as having *spoken.* This is a rather more discrete query, because not every elected member actually speaks when they are in office, and some of those who spoke might have spoken so little that the clerks recording speeches took no notice of what they said.

Because recorded parliamentary speech is an index of importance, the members who were recorded as having spoken before and after 1832 offers a list of those members who were important enough to have taken notice of the public in both periods. This is a good proxy for which members mattered in the public imagination *and* met the criterion for having been elected both before and after the reform. Do we imagine this number to have been large or small?

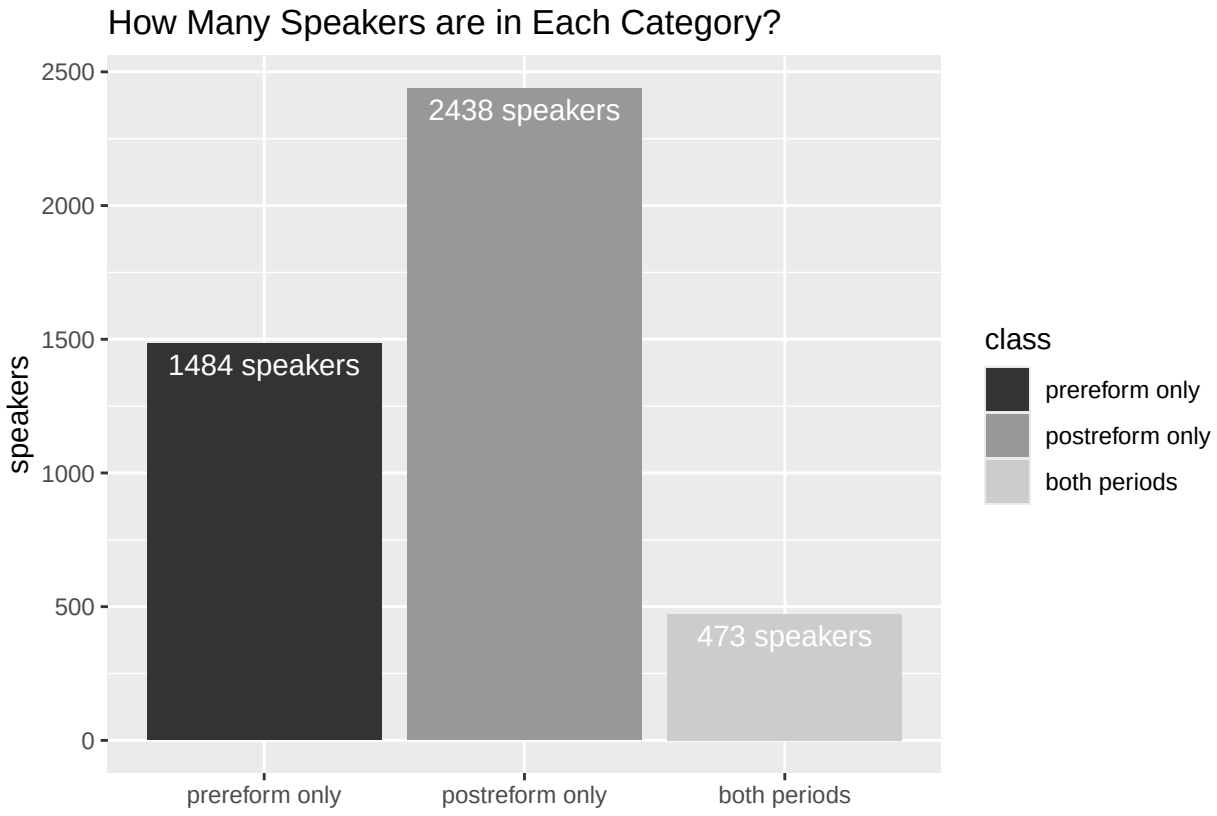
We might guess that it was vanishingly small, because a reform of historical significance could have swept away all incumbents. On the other hand, many historians of the 1832 Reform Bill have pointed out that it didn't shift the election process so drastically as had once been assumed.

In any case, to identify the speakers who appear in both the prereform and postreform datasets, we will use another function – inner_join() – to identify what is known as the "intersection" of the two datasets. That is, we are looking for speakers who occupy a row in both the *prereform_speakers_wordcount* data frame and the *postreform_speakers_wordcount* data frame. The syntax for inner_join() is just like that of anti_join(), but the result is to *keep* rows that appear in both datasets, not to exclude those rows.

```
## Joining with `by = join_by(suggested_speaker)`
```

```
## [1] 473
```

We now have actual numbers for the members of parliament recorded as only having spoken before the reform, those who were recorded as only having spoken after the reform, and those who were recorded as having spoken both before and after the reform. Let's represent our results visually.



Certain caveats are important for interpreting this number. While the speaker data gives us reliable insights into relative trends in membership and speech, the numbers are only so good as our dataset.

One of the great advantages of text mining is that we can analyze not just the speakers in parliament, but the consequences of the 1832 Reform Act for what was spoken about in parliament. Let's begin by asking what parliament spoke about before and after the reform.

Working With Word Count

```
## Joining with `by = join_by(suggested_speaker)`
```

```
## Warning in left_join(., all_classes): Detected an unexpected many-to-many relationship between `x` a
## i Row 4 of `x` matches multiple rows in `y`.
## i Row 3923 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
## # A tibble: 20,775 x 5
##    suggested_speaker        year wordcount period    class
##    <chr>                   <dbl>     <int> <chr>     <chr>
##  1 abraham_hume_1561        1826       702 prereform prereform only
##  2 adolphus_dalrymple_1968  1831       457 prereform both periods
##  3 adolphus_dalrymple_1968  1832      1699 prereform both periods
##  4 alexander_baring_1475    1830      5671 prereform prereform only
##  5 alexander_baring_1475    1830      5671 prereform prereform only
##  6 alexander_baring_1475    1831      7135 prereform prereform only
##  7 alexander_baring_1475    1831      7135 prereform prereform only
##  8 alexander_dawson_2455    1826      1935 prereform prereform only
##  9 alexander_dawson_2455    1826      1935 prereform prereform only
## 10 alexander_dawson_2455    1826      1935 prereform prereform only
## # i 20,765 more rows
```

Note that we have created two columns that annotate the speeches and speakers in different ways. The "period" column refers to the year in question – were the n words said in 1831 prereform or postreform? The "class" column sorts the speakers themselves – did this speaker only speak in the prereform years, the postreform years, or in both periods?

We also have a *wordcount* column that reflects the words each speaker spoke per year.

Next, let's sort the data to find the speakers who spoke the most per year within each category – prereform, postreform, and continuously through the periods.

To do this, we'll want to first create a new column, *wordcount_per_speaker*, which adds together each speaker's total wordcount. If we have multiple "wordcount" columns, we should add information about what each refers to. To make it less confusing, we'll rename our previous *wordcount* column *wordcount_per_year*. It's good practice to keep renaming fields in our datasets for clarity so that we don't get confused about what they mean. To calculate *wordcount_per_speaker*, we'll use the *group_by()* and *mutate()* functions that we've seen before. Note that we won't use *summarize()* in this case because we still want to keep the *wordcount_per_year()* field upon which the new count is based.

we'll use *group_by()* to group the data by "class," then use the *arrange()* and *slice_head()* functions to take the top wordcounts for each category.

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

```
## # A tibble: 15 x 4
## # Groups:   class [3]
##    class           suggested_speaker      wordcount_per_speaker top
##    <chr>           <chr>                                  <int> <lgl>
##  1 both periods    robert_peel_1664                     1982866 TRUE
##  2 both periods    henry_brougham_1679                  1616327 TRUE
##  3 both periods    joseph_hume_1712                     1336979 TRUE
##  4 both periods    john_russell_1885                    1046022 TRUE
##  5 both periods    daniel_oconnell_2552                  811831 TRUE
##  6 postreform only frederick_shaw_2644                  2117630 TRUE
```

```
##  7 postreform only john_roebuck_2850                  1935816 TRUE
##  8 postreform only henry_ward_3175                    1815870 TRUE
##  9 postreform only thomas_wakley_3333                 1647616 TRUE
## 10 postreform only george_howard_2466                 1587512 TRUE
## 11 prereform only  william_huskisson_1230             1765120 TRUE
## 12 prereform only  george_canning_1114                1333782 TRUE
## 13 prereform only  james_mackintosh_1886              1239952 TRUE
## 14 prereform only  henry_herbert_734                   660530 TRUE
## 15 prereform only  robert_wilson_2106                  419085 TRUE
```

Annotating Our Data

Before we go any further, let's do some work to make the speaker names more elegant. Because of the structure of our data, names that were algorithmically cleaned are labeled with a standardized name and number, e.g. "robert_peel_1665." When we make a graph to share, we'd like to substitute one of Robert Peel's names as given in the text for this awkward string. The names as given in the text are in the *hansard* dataset under the *speaker* column. We can grab any one of them – minus the titles like "Lord Chancellor" with which some speakers are referenced – and it will read like a normal name. The *str_detect()* function is useful for searching for a string. In this case, *!str_detect()* will allow us to throw out names that include titles we don't want to use. The *sample_n()* function is useful here as it tells the computer to select a random pick from a set; in other words, if we've grouped Robert Peel's many possible names, and we have the set "Robert Peel," "Sir Robert Peel," and "Mr. Robert Peel," the computer is instructed to pull out one of these at random.

After we choose one name to use from the full Hansard list of literal speaker references, we can use these names to annotate the previous dataset, *all_speakers_annotated*, which has the *wordcount* (per speaker per year) and speaker class.

Let's try it.

```
## # A tibble: 5 x 2
## # Groups:   suggested_speaker [5]
##   speaker           suggested_speaker
##   <chr>             <chr>
## 1 Mr. Edmund Peel   edmund_peel_2792
## 2 Mr. Jonathan Peel jonathan_peel_2471
## 3 Sir R. Peel       robert_peel_1664
## 4 Sir R. feel       robert_peel_1664|robert_peel_4101
## 5 Mr. W. Peel       william_peel_1975
```

The results may not be impressive for the vast majority of speakers who spoke very little. But for major speakers like Robert Peel, this *speaker_key* dataset will help us reconcile the many names with which they appeared in the text with one numeric id number, then reconciling that numeric id with one recognizable name from the Hansard transcription.

Using Left Join to Annotate the Data

We've previously used set theory and "joins" to find the intersection and difference between two datasets. Here, we'll use another kind of join – the "union" – to annotate one dataset with information from another dataset.

That is, in *all_speakers_annotated*, we have a list of speakers and how many words they spoke every year. If we use *left_join()* to merge that dataset with the dataset *top_speakers()*, we'll have a record of whether each speaker was one of the most prolific speakers in their category. If we use *left_join()* to merge the dataset again with the dataset *speakers_key()*, we'll be able to use the literal transcriptions of speaker names to annotate the visualization later.

The function *left_join()* takes as its first argument the name of the dataset with which we're merging our data. A second optional argument is the "by" argument, which allows us to specify a column or set of columns to use for matching. In the case of the first join, we'll specify two fields, concatenated into a list, using the syntax *by = c("suggested_speaker", "class").*

```
## Joining with `by = join_by(suggested_speaker)`
```

speaker	top	year	wordcount_per_speaker_per_year	class
Mr. Edmund Peel	NA	1831	2654	both periods
Mr. Edmund Peel	NA	1832	23	both periods
Mr. Jonathan Peel	NA	1831	34	prereform only
Sir R. Peel	TRUE	1823	27786	both periods
Sir R. Peel	TRUE	1824	46554	both periods
Sir R. Peel	TRUE	1825	35058	both periods
Sir R. Peel	TRUE	1826	42461	both periods
Sir R. Peel	TRUE	1827	37982	both periods
Sir R. Peel	TRUE	1828	108757	both periods
Sir R. Peel	TRUE	1830	145546	both periods
Sir R. Peel	TRUE	1831	124646	both periods
Sir R. Peel	TRUE	1832	86337	both periods
Mr. W. Peel	NA	1823	390	both periods
Mr. W. Peel	NA	1824	924	both periods
Mr. W. Peel	NA	1825	215	both periods
Mr. W. Peel	NA	1826	609	both periods
Mr. W. Peel	NA	1830	494	both periods
Mr. W. Peel	NA	1831	1660	both periods
Mr. W. Peel	NA	1832	855	both periods
Mr. Edmund Peel	NA	1832	71	both periods
Sir R. Peel	TRUE	1832	32476	both periods
Sir R. Peel	TRUE	1833	108710	both periods
Sir R. Peel	TRUE	1834	103120	both periods
Sir R. Peel	TRUE	1835	85622	both periods
Sir R. Peel	TRUE	1836	127812	both periods
Sir R. Peel	TRUE	1837	116960	both periods
Sir R. Peel	TRUE	1838	125225	both periods

speaker	top	year	wordcount_per_speaker_per_year	class
Sir R. Peel	TRUE	1839	168503	both periods
Sir R. Peel	TRUE	1840	112914	both periods
Sir R. Peel	TRUE	1841	137861	both periods
Sir R. Peel	TRUE	1842	208536	both periods
Sir R. feel	NA	1841	148	postreform only
Sir R. feel	NA	1842	48	postreform only
Mr. W. Peel	NA	1832	439	both periods

Visualize the Results

Sometimes the most effective way of analyzing data is to "inspect" the results of counting, as we have done henceforth. For simple results, like a list of names or words, we prefer a table.

But for questions about information when several dimensions of data are related to each other – for instance speaker words over time when speakers have been categorized in three "classes" – visualization may be useful for conveying how many factors are related to each other. Next, we'll use *ggplot()* to create a visualization of the results as a timeline.

In the case of the data about speakers before and after 1832, we have rich information that deserves to be represented in an interesting way. We have three categories of speakers – prereform, postreform, and continuous speakers – and information about how many speakers are in each category. We also have information about how much each speaker spoke each year. Can we model this information together and learn something from the result?

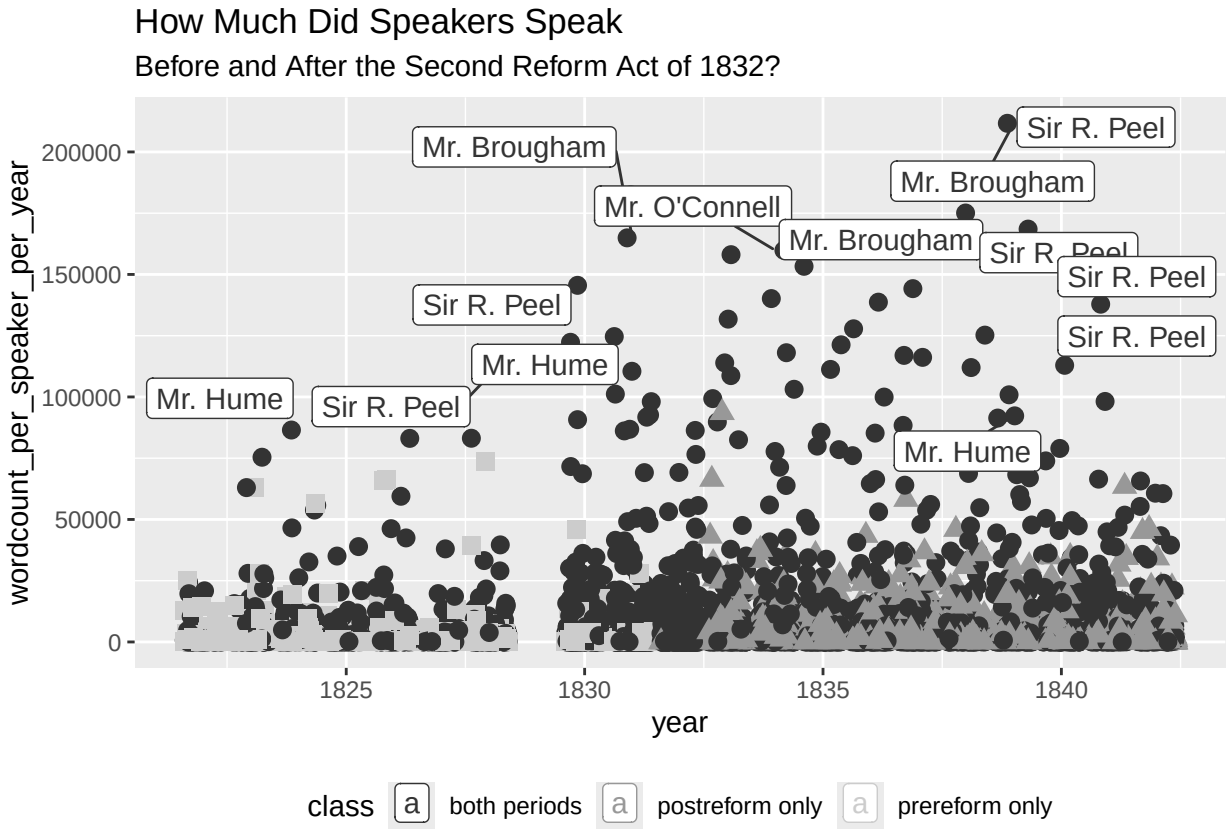
in the code block below, we'll create a "dot plot," mapping one point for each of the 10,810 speakers in the data frame. We will use the color and shape of the dots to map the three classes of speakers – prereform, postreform, and continuous.

Contingent labeling

Because labeling all the speakers would result in an unreadable graphic, we will tell the computer to only label those speakers who are classified as "TRUE" in the column *top*, where we annotated if a speaker was among the top n speakers of each class by wordcount. The syntax for this operation is given inside the parentheses of *geom_label_repel()*, where the coder has an opportunity to point to a new dataset as the source of the text in the labels. Here, we tell the computer which dots to label and which to leave blank with the use of *subset(graph_ready, top == TRUE)*. Operations of this type are relatively advanced for visualization, but they are extremely useful for the kind of dense, text-forward information that implementors of text mining often want from their analysis.

If you are new to visualization with *ggplot()*, we encourage you to play around with the code below to understand how the components work. Try changing the size of the dots by adjusting the "size" parameter inside *geom_jitter()*. Or try changing from a *geom_jitter()* plot to a *geom_point()* plot. For an in-depth tutorial about the powerful *ggplot()* library and the philosophy of the "grammar of graphics" behind it, we encourage you to consult Hadley Wickham, *A Layered Grammar of Graphics,"* Journal of Computational and Graphical Statistics,* vol. 19, no. 1, pp. 3–28, 2010.

```
## Warning: ggrepel: 164 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



********* HEY STEPH – Lord Brougham shows up as both a "both periods" speaker and a "postreform only" speaker, presumably because of a data quality issue. Can you have a look at this and help make sure that the resulting chart doesn't have errors? And/or maybe include a module on correcting Brougham by hand?

What do we learn from this plot? A great deal. One is about the relatively higher counts of speeches in the 1830s than the 1820s, irrespective of the reform Act. Speeches by lord Althorp, Robert Peel, and Henry Brougham regularly reached into range of 150,000 to 200,000 words – which if delivered at breakneck speed would have taken a stunning *fifteen hours* to deliver.

Even the speeches of lesser speakers were longer in the new decade, stretching towards 50,000 words instead of 20,000 words as the upper limit for most rank-and-file speakers – meaning that after the reform, multiple speakers each year addressed the house for *five hours* or more at a time.

We also learn that by and large it was the continuous speakers – not the prereform-only aristocratic appointees or the postreform-only members elected by the middle class – who spoke at the greatest length. Represented with black dots, these men included prime ministers like Robert Peel.

We can also use this visualization to note the few prereform-only speakers who delivered relatively long

speeches, for instance Mr. Canning and Mr. Huskisson, the latter important to the Board of Trade, represented by light gray squares.

Equally exceptional were the few members of parliament elected only after 1832 who spoke with prominence, among them John Roebuck and Frederick Shaw. These men whose careers were endorsed by reform swam in a sea of seasoned representatives who had first joined parliament in an earlier era.

Note that our visualization is only as good as the quality of the data. Where speakers have not been reconciled correctly – as is the case with "Lord Brougham" (classified as postreform only), who in reality is the same as "Henry Brougham" (classified as a continuous speaker) – they may appear as discrete speakers when indeed they are the same.

Exercises

1) As we discussed in the beginning of this chapter, analyzing the individual or collective speeches of men in Parliament can give insight into the workings of the Parliament as an institution of power. We can explore the roles of elite white men as they exerted their will–and the will of their peers– through the Parliamentary institution. We can also explore the roles of women in Parliament and examine how women differ in their contributions, as they were prohibited from becoming members of Parliament. Further, the woman's presence had to be mediated by one of the MPs, as they could not be there without being selected or approved. In result, women were treated as instruments of power in Parliament.

We can further explore how the role of women manifested in Parliament. First, count the total number of words women spoke in Parliament. Refer to the table below, which visualizes the women in Parliament and the year in which they spoke. Can you find a pattern across the kinds of topics women are brought to Parliament to speak on?

Name	Year	Name	Year
Mrs. (Mary Ann) Clark	1809	Ms. Cunninghame Graham	1887
Mrs. Bridgeman	1809	Mrs. Dillon	1902
Miss Mary Ann Taylor	1809	Mrs. M'Govern	1902
Louisa Demont	1820	Mrs. Mitchel-Thomson	1907
Franchette Martigner	1820	Ms. Cave	1907
Mrs. Walrand	1824		

Note: The way in which their names are recorded in this table might not reflect the way in which their names appear in the debates because the names have been consolidated into one representation

2) Use the techniques shown above to move between a distant reading of the top words spoken by Mrs. Walrand or Louisa Demont in all of her speeches to the full debate. (As a reminder, you can view all of a speaker's speeches by filtering for just that speaker. You will also need to load the data for the appropriate years.) How does transitioning between the different modes of reading text provide insight into this specific debate? How does the data frame illuminate the role of women in 19th-century Parliament?

3) Adjust the above code to search for a different office title. What can you infer about the different MPs who held this title? Can we support our inferences through close readings of their speeches?

To get you started, here are a few office positions: - Chancellor of the Exchequer - Prime Minister - Attorney General - Lord Chancellor

4) At the beginning of this chapter, we compared counts are based on the "speaker" column with counts based on the "suggested_speaker" column. Expand each table to show 30 rows. How are the two tables different? What are some pros and cons about visualizing the original speaker names and the disambiguated speaker names? Identify a future research question based on what you learn.

5) Find the speakers who were referred to by the greatest variety of names in the 1830s. Share the results. Investigate; what characteristics do you think they had in common? AI's are great for tasks involving linked data — that is , background about individuals — especially for the individuals who are well known and who had published obituaries and biographies, like many members of parliament. Feed the resulting list to an AI and ask it for input about what they had in common. What did you learn?

6) Find the speakers who held the highest number of offices in the 1830s. Share the results. Then, investigate with AI: what do these individuals have in common? Tell us what you learned.

7) We've used a list of speakers' words over time to investigate many aspects of the data, but let's go further. For the continuous speakers, almost everyone talks more after the reform. But for whom is the difference how much they spoke pre reform and post reform the greatest? Share your results. Again, use AI to investigate what those speakers have in common. Tell us what you learned.

8) Use your code to identify the continuous speakers who have the greatest difference between the year during which they spoke the most and the year during which they spoke the least. Share your results. Again, use AI to investigate what those speakers have in common. Tell us what you learned.

9) Rework the final visualization so that what is labeled are not the top speakers overall, but the top speakers in the prereform and postreform categories. Show your results.

Table 3.3: Excerpt of Debate Text (First 20 Sentences)

text
rose, and addressed the House to the following effect:—
; I confess that, in bringing before this House the question on which I now rise to address you, I feel not a little disheartened by the very intense interest excited in the country, and the contrast presented to those feelings by the coldness which prevails within these walls.
I cannot conceal from myself, that, even in quarters where one would least have expected it, a considerable degree of disinclination exists to enter into the discussion, or candidly to examine the details of the subject.
Many persons who have, upon all other occasions, been remarkable for their manly hostility to acts of official oppression, who have been alive to every violation of the rights of the subject, and who have uniformly and most honourably viewed with peculiar jealousy every infraction of the law, strange to say, on the question of Mr. Smith's treatment, evince a backwardness to discuss, or even to listen to it. Nay, they would fain fasten upon any excuse to get rid of the subject. ""
What signifies inquiring, "" say they, ""into a transaction which has occurred in a different portion of the world?""
As if distance or climate made any difference in an outrage upon law or justice.
One would have rather expected that the very idea of that distance; the circumstance of the event having taken place beyond the immediate scope of our laws, and out of the view of the people of this country; in possessions, where none of the inhabitants have representatives in this House, and the bulk of them have no representatives at all, one might have thought, I say, that, in place of forming a ground of objection, their remote and unprotected situation would have strengthened the claims of the oppressed to the interposition of the British legislature.
Then, says another, too indolent to inquire, but prompt enough to decide, ""It is true there have been a great number of petitions presented on the sub-
* From the edition published by Hatchard and Son, with the sanction of the London Missionary Society.
ject;
but then every body knows how those petitions are procured, by what descriptions of persons they are signed, and what are the motives which we know influence a few misguided, enthusiastic men, in preparing them, and the great crowd in signing them.
And, after all, it is merely about a poor missionary!""
It is the first time that I have to learn that the weakness of the sufferer; his unprotected situation; his being left single and alone to contend against power exercised with violence—constitutes a reason for this House shutting its ears against all complaints of those proceedings, and refusing to investigate the treatment of the injured individual.
But, it is not enough that he was a Missionary; to make the subject still more unpalatable, for I will come to the point, and at once use the hateful word, he must needs also be a Methodist.
I hasten to this objection, with a view at once to dispose of it.
Suppose Mr. Smith had been a methodist; what then?
Does his connexion with that class of religious people, because, on some points essential in their consciences, they are separated from the national Church, alter or lessen his claims to the protection of the law?
Are British subjects to be treated more or less favourably in courts of law; are they to have a larger or a smaller share in the security of life and limb, in the justice dealt out by the government, according to the religious opinions which they may happen to hold?
Had he belonged to the society of the methodists, and been employed by the members of that communion, I should have thought no worse of him or his mission, and felt nothing the less strongly for his wrongs; but, it does so happen, that neither the one nor the other of these assumptions is true: neither the Missionary Society, nor their servants, are of the methodist persuasion.

The Top Ten Words Spoken in Prereform and Postreform Parliament

word	wordcount
prereform	
house	59,045
hon	58,821
country	38,776
noble	34,335
bill	32,251
government	27,383
lord	25,402
time	21,347
question	20,757
subject	20,726
postreform	
house	121,241
hon	119,156
noble	79,627
bill	66,044
lord	65,073
country	55,848
government	54,333
question	41,455
ireland	40,072
subject	38,767