**To Do Next:**

* Identify R packages that will help me the most with all of this stuff
* Look into how text mining algorithms deal with messy data coming in
  + How do you take away ‘the fluff’ from a bunch of different types of files?
* See if there is anyone in Auckland doing this type of research
* If the data is in the right form, what can we do? What data can we get into the right form?
* Shortened document for Nick

**Things to Think About**

* Address things in red, also tease apart problem ideas from solution ideas
* Add some more information about *why* you have to do things
* Audience won’t actually be writing code to clean etc so we need to think about what types of data we can actually use here
* Create a system that guides someone through the process but they have to make some decisions
* Be super clear with what we are *actually* doing and then what you *could* do beyond that
* How do we make this easy to use with things that young people care about?
* How big of a gap between raw data and analyzable data is manageable?
* What types of things are feasible to try and do?

**Project Notes**

* Some of these books/chapters were written before the tidyverse was a thing so I probably won’t write code in the way that they do → really only take conceptual things from these
* Most of this is tidying the data and making sure its in a usable form
  + Tokenization
  + Removing stop words and extra nonsense (publishing info, headers, etc)
  + Is it tidy? Does what you’re trying to do require tidy data or would a different form be more useful for analysis/graphing?
  + *Be careful about removing things too quickly → stop and think about what you are actually trying to do and what you need to do to get there*

**Questions to Address**

**Big Picture**: What exactly is text mining used for? Why do people try to do it?

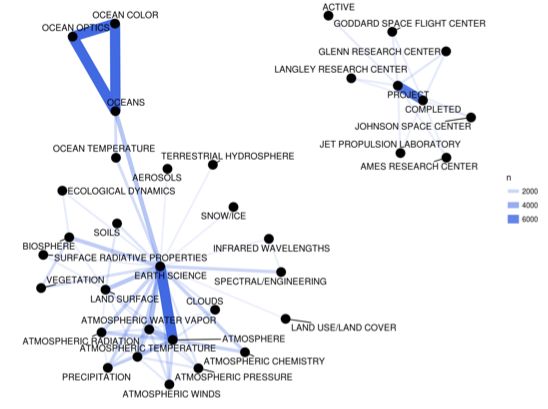
* Task: Add more examples and explanations of what these things actually mean
* Keep in mind: Could become a little preamble to the module
* Keep in mind: Hover over feature → get what you want when you need it
* Overall, text mining is about taking some sort of information from unstructured text data and do things like the following:
  + You can turn text into something that appears more like data as we know and can quantify aspects of it (word frequencies, correlations between words, networks of words that belong together, words that can be assigned to emotions)
  + For example, what are Jane Austen’s most commonly used words in her novels? Were those popular throughout all of her novels or just a select few? Did her writing style change over time? How can we assign the words she used into meaningful insights about language?
  + Another example: Can you take the transcription of a speech and determine what political party the speaker belongs to? Can you extract the feelings and emotions written into the text?
  + Can we use the way that people write about things in emails and news articles to predict things like stock market crashes and disease outbreaks?
  + NASA example: Humans read over NASA dataset descriptions and then determine which ‘keywords’ they should attach to the dataset as a short summary of what it pertains to → what if you could build a text mining program that does this for you? Can you relate words into a network in some way that benefits the human user?
* Want to measure how words are used in similar/different ways
  + Can we find patterns in how certain words are used? Are words used differently between different groups of people? What types of words are relevant to what we care about? Is there anything novel about the way someone uses a word?
* How do documents compare to each other? How do they change over time?
  + For example, one book looked into how the sentiments in Jane Austen’s novels change from chapter to chapter
  + You could look at the course of an author’s life and see how their writing styles changes
  + You can look at new articles regarding something or someone like Louis CK, for example. I can imagine that following the sexual assault story surfacing, the words associated with “Louis CK” not only themselves changed, but the sentiments of the words changed as well
* Text data is unstructured and lacks clearly defined variables
  + Not many people know what to do with that
  + If we can figure out how to turn it into a digestible set of words, we can do analysis that can find things like relevance, importance, and uniqueness in single words, groups of words, sentences, etc
  + Essentially turns text into something than can be statistically analyzed (which is obviously what most statisticians feel comfortable working with)
* Text data describes some object, carries some point of view, etc
  + We can draw information from chunks of text
  + For example, we can analyze reviews on Amazon for a specific product
    - Can find the words most commonly used to describe that product
    - Can find the general sentiments about the product
    - Can use the analyses you run to predict what information new reviews carry
  + Going back to the political speeches, you can use text to discern what point of view someone may carry in regards to a specific subject

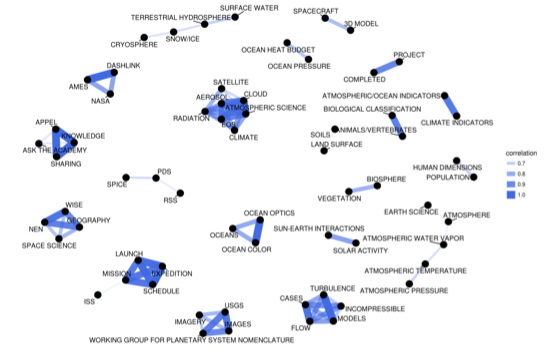
**What are the basic problem types that text analysis people are trying to do?**

* Task: Need to distinguish between types of problems and steps along the way
* The basic problems here start with what you’ll actually do with the data.
  + Will it be a summary of one or multiple documents?
    - This can include comparisons between documents and within one document itself.
  + Will it be using data to make predictions about other documents?
    - Ex: Based on the text of an Amazon review, can we predict how many stars the review has?
* Things text analysis can be used for:
  + Security measures: Could it flag blogs, emails, etc that seem to be threatening?
  + Customer satisfaction: Can we take data collected from customers and use it to modify business strategy and products?
  + Search engines: Can we make search algorithms more efficient? What is the best way to filter through text to find ‘the most relevant’ information?
  + Sentiments: Can we teach a computer to pull emotions from words on a page?
  + Humanities research: Can we make the process of reading through documents more efficient?
    - Could be spending much less time simply searching for things relevant to your research
    - Build a strong network of documents related to your topic of interest
      * I’m thinking here about an academic with a narrow research interest → can we expedite the literature review process?
  + Biases: Can we find gender and racial biases in text? How do you quantify that?
  + Analyzing patient or customer comments: Can we extract the most meaningful points in a patient’s description of their symptoms? Can we determine the problem based on a customer’s comments in an insurance claim?
* It honestly seems to me that the entire point of this is to take the massive amount of qualitative text data that we have and translate it into numbers that we can process and understand relatively quickly and easily.
  + I.e. it takes *way* too long to read through lots of text and decipher its meaning!
  + It would be nice to have programs that derive trends and patterns in your corpus of documents → makes it easier to narrow your scope more quickly if you’re doing research OR to make things like strategic business decisions without having to read through thousands of reviews

**Steps along the way...**

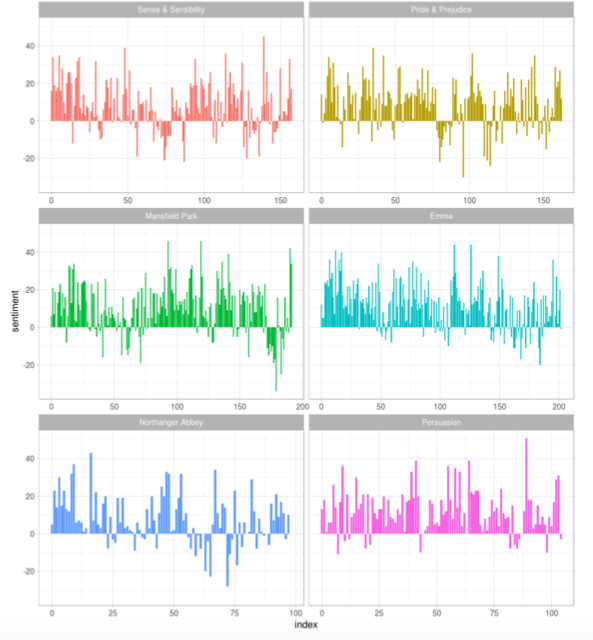
* Tokenization
  + *Absolutely* *must be done* as a step towards changing text data into a structure that we can work with it in
    - Key step for changing from unstructured to structured data
  + Break a string into parts
    - Single words
    - N-grams → multiple word chunks
      * For example, breaking a text down into bigrams would look something like this → “Cassidy went to the park” would break into “Cassidy went” and “went to” and “to the” and “the park”
    - Full sentences
  + Can use regex (regular expressions) to denote the pattern you’d like to split on (probably a space in many cases)
  + This is the step where things like punctuation and stop words are removed
  + Tokens are the items fed into further analysis
* Creating document-term matrices
  + A necessary step in order to do any analyses
    - Most things require you to have a matrix in this form
  + Each row is a document (book, article, etc), each column is a term, each value is the number of appearances of that term in that document
  + These are very sparse! If your collection of documents doesn’t have much overlap in terms of word use, the matrix will have *many* zero entries
  + Not sure how to provide an example of this → they are relatively easy to create once you’ve got your tokens
* Tf-idf Statistic
  + *This is a type of problem → it is a “tool” that is commonly used to help you complete your analysis or decide what is important to focus on*
  + Stands for term frequency-inverse document frequency
  + This basically calculates the weights (importance) of words in a document or a corpus
  + Decreases the weight for common words and increases the weights for words that aren’t used much in a collection of documents but that are important to some documents
  + Can find tf-idf between multiple things (can be between documents or even within different sections of a document as demonstrated in the NASA case study down in my notes)
    - Can see section 3.4 and 8.3 of tidytext book for nice examples
  + Task: What do people do once they find the ‘important’ words?
    - In the case of finding relevant documents, you search for a keyword and the algorithm returns the documents where that keyword has a high tf-idf score
    - In the NASA case study, they could’ve used the dataset description terms with the highest tf-idf scores to help assign keywords to the description
* Correlations
  + Same type of thing as with numeric variables although you look at a slightly different thing (a phi coefficient) for binary outcomes (will the words occur together or not)
    - Section 1.5 in tidytext did a correlation test for word frequencies between Jane Austen, the Bronte sisters, and HG Wells
  + Pairwise correlations
    - Note: need to turn data into wide matrix to do this (widyr package)
      * Widens data, does correlation, retidies data
    - This is a good thing to use because it explores *relationships* between words rather (how often they appear together versus not appear together) than just how many times they occur together (more reflective of how many times the individual words occur)
  + Can make really awesome network graphs using word correlations (section 4.2.2)
  + Note: co-occurring measures and correlation measures are slightly different things
    - The first figure is co-occurring keywords in NASA data and the second is correlated keywords



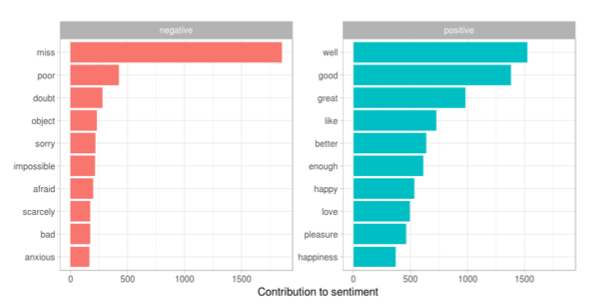


**Types of analysis...**

* Sentiment analysis
  + Can quantify positive/negative attitudes of texts
    - Why do this? Useful for analyzing trends in opinions (via social media or what have you), customer satisfaction
  + There are many different lexicons each with unique uses
    - Some lexicons just determine whether a word has a positive or negative connotation
    - Some lexicons assign numeric scores to words (for instance, the AFINN lexicon assigns scores ranging from -5 to 5)
  + N-grams are slightly more nuanced and give more information about relationships between words
    - For instance, using an n-gram would pick up the fact that “I had no problem with it” is actually a positive sentence rather than negative due to the single word “problem”
    - It is hard for machines to understand the nuances of human language!
      * It has little to no sense of sarcasm and tone
  + How do sentiments change over space or time?
    - For example, a company could track how their commercial is being perceived by the public through Twitter.
    - The following graph shows Jane Austen novels split into 500 word chunks and the overall sentiments for each chunk



* + How does each word contribute to a document(s)?
    - You can see how much weight a word carries (see next graph)



* + Keep in mind: Could compare sentiments between groups
* Frequency analysis
  + Why do this? A great way to get a feel for your data, you can see some overarching themes and words to be looking out for
  + Pretty simple overview of the data → count word frequencies to find most common words (can do this by sentiment, document, etc)
    - For example, compare the most commonly used words for the #auckland and #wellington hashtags on twitter
  + Find word associations by designating a word like ‘Cassidy’ and finding the words most correlated with it or most commonly used with it
* Text Classification
  + Assigning a text to one or more categories
    - Can do this as a summary measure OR as a prediction
    - Could classify with the help of a human guiding the process so that the algorithm can learn
    - Could also classify by topic modeling or clustering (unsupervised)
    - Can do things like determine the genre of a text by using this (is it comedy, horror, drama, etc)
    - Sentiment analysis and topic modeling are types of text classification
  + To be able to classify a document as a specific ‘type’ or ‘topic’ allows us to summarize it nicely
    - It’s nice to know what umbrella topic you’re going for in order to narrow your search (i.e. finding a specific book in the library)
* Topic modeling
  + This is machine learning-esque in that it does unsupervised classification of documents into clusters
    - As far as I’ve seen, you specify the number of groups you’d like
      * Can test different numbers to see which ones make the most sense
  + Every document is a mixture of topics and each topic is a mixture of keywords; can find probabilities that a word is in a topic or a topic is in a document
  + Uses: finding patterns in text, finding shifts in topic in texts, put similar documents together
  + All of Chapter 6 in TidyText is about this topic
  + Chapter 8 (the NASA case study) also does a nice example with this
    - They broke NASA datasets into 24 topics (after testing a bunch of different values) and then looked at the top ten words per topic to see if they made any sense → it was clear that the algorithm had put similar datasets into the same groups → there were groups on atmosphere, oceans, finance, technology, etc
      * It actually turned out that the topic it selected often corresponded to the ‘keywords’ for each dataset which was cool to see

**Putting analysis to use...**

* Examining one document
  + What terms are important to this document?
    - Word frequencies
    - Tf-idf score
  + Can it be parsed down into different themes?
    - Clustering
  + Did the writing style/attitude change over the course of the document?
    - Sentiment analysis
  + Can you break it into sections and evaluate differences?
    - Topic modeling
* Comparing multiple documents (a corpus)
  + Using collections of documents to look at similarities and differences between individual documents
  + Using tf-idf to explore relationships between documents (whether that be emails, different parts of a dataset, articles, books, etc) and to explore relevance to certain keywords
  + Can you look at the differences between documents?
    - Can run things like correlation tests
    - Sentiment scores and visualizations
    - Do they use different words to talk about the same broad topic? Do they talk about completely different topics?
  + You can compare word usage between documents or things like different twitter users by calculating log odds ratios (was a word that was used more likely to have been said by person A or person B?)

**What does the data look like? What kinds of data are used and what does the data look like after processing? Keep in mind: This is what we need to think about in terms of what data we can accept.**

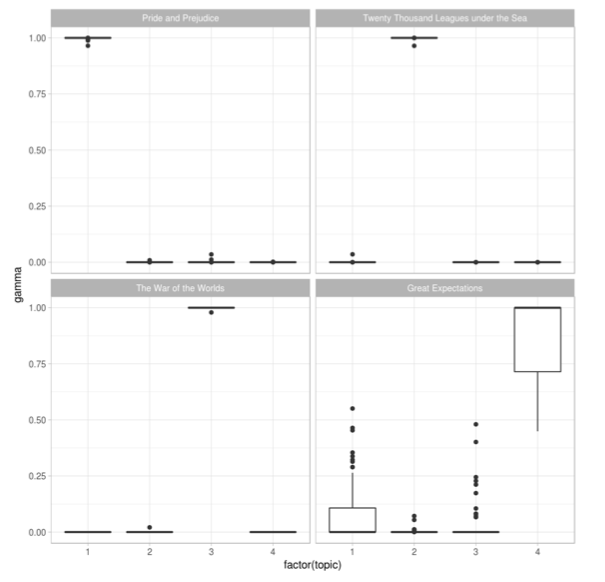
* **This is something to think hard about when we are actually designing the program**
  + **What types of data can we accept? How narrow will our scope have to be?**
  + **You can remove a great deal of “extras” with a few simple commands in the tm package (stripWhitespace, removeNumbers, removePunctuation, etc)**
* Both beginning and end vary but you always remove a bunch of stuff in the beginning
* First of all, what types of documents will you be using? How do you plan on breaking it down (tokenization)?
* Unstructured data: there aren’t really variables → we can put it into ‘tidy’ format at first
* After the data is ‘tidy’, it often has to be put into a matrix to do any actual analysis on it
* Relatively easy to switch back and forth between tidy and matrix formats
* Sometimes data needs to be restructured a bit to make nicer graphs

**What sorts of displays/visuals/pictures do they create?**

* Histograms and Bar Plots
  + Word counts → simple graphs that show which words are used the most
  + Sentiment scores
    - Section 5.1.1 has nice example
    - Plot things like which words carry the most influential sentiment score
    - Can also use histograms to show sentiment scores over the course of a book (could break it down into chunks of, say, 300 words each and examine the sentiments for each chunk) → example of this was done on a Jane Austen novel (it was clear where events like deaths and break ups took place in the novel)
* Line graphs
  + Trends over time
  + Can look at the usage of specific words in things like tweets over time
    - Book did this and it was interesting to track because the woman originally used her Twitter for personal life but then switched to using it a lot for data science tweets, so she had some nice line graphs showing the evolution
* Networks
  + Groups → are there natural clusters in the words used together in documents
  + Correlations → can visualize words that are correlated with each other
  + See above for some network plots
* Scatterplots
  + Comparing word frequencies between two things
  + Fit an identity line → words that fall close to or on the line are used in similar frequencies between, say, two different authors
    - Example of this would be to compare the word usage between Jane Austen and Emily Bronte → which parts of their language are the same? Which are different?
  + Words that fall far from the line are more heavily used by one author
  + Section 1.5 has a nice example of this being done



* Wordclouds
  + These make nice visuals of the most commonly used words
  + Can be faceted by something like sentiment → cloud for most common positive words, cloud for most common negative words
  + 
* Boxplots
  + Mostly in the topic modeling sections → plots gamma (per-document-per-topic probabilities task: explain more and get examples for this) against the topics (how well did it do!!)
    - Library example → the boxplot shows how often the chapters in each book were assigned to one topic (which, based on the algorithm they ran, should be the book itself)
    - Gamma should give you an idea of the proportion of words from a document that are from a specific topic
  + Also did a nice little heat map in that section



**How easy are these things to understand?**

* I honestly think that the hardest part is getting the data in the correct format and knowing what you actually have to do
  + Interpreting results isn’t super tough with these
* May need to do a bit more research if I get into the machine learning aspects of it but I can probably handle the wrangling and cleaning pretty well
  + Keep in mind: Could include these types of things in a paragraph about what you *can* do with this, but we won’t actually do it

**What do you have to know before you can do them?**

* You need to be super clear with what you’re trying to do
  + What is the point of working with the text data you have? What types of insights are you hoping to gather from it?
* Need to know the proper format to put your data into
* Know your data
  + What things are important to keep in mind?
  + Unique stop words?
  + Different lexicons for sentiment analysis?
  + What’s your end goal?
    - Are there methods discussed here that will help you get there?
    - What combinations of analysis would you have to use to accomplish what you’d like to do?

**Glossary Words**

* **STILL NEED TO DO**: Add definitions and more words obviously
* Tokenization: process of splitting text data into pieces that can then be analyzed
* Tf-idf
* Corpus: a collection of documents
* Gamma
* Document-term matrix: where you store token occurrences across a set of documents
* N-gram: a token that is multiple words long → a bigram is a two word token
* Regex
* Stop word: words that are removed before analysis such as common English words like ‘and’ ‘the’ ‘or’
  + Can also remove more words based on the situation: if you are analyzing movie reviews, you can remove the word ‘movie’ because it’s rather meaningless in that context

**Ideas for the Module**

* Some description of text mining as a whole
  + What is it?
  + What you can do with it?
  + Why you would want to do those things with it?
  + What we’ll actually do in the module
  + Things that could be done beyond the module
    - Lots and lots of machine learning
* To think on…
  + What decisions do we want the user to have to make?
  + What types of data will be usable?
  + Which types of analyses do we want to demonstrate with it?

**Research Notes**

**Modern Data Science with R**

**Chapter 15: Text as Data**

* *Regular expressions* (regex): used to answer questions about the text (how many times does the character Macbeth speak in Shakespeare plays?) à do this by finding patterns
  + grep() is the function we use to do this à specify pattern and then the vector in which you’ll find the pattern
  + grepl() is a slightly different thing à returns a logical vector as long as the haystack à i.e. it gets the patterns and will be longer than just grep
  + *metacharacters*: ‘.’ will return any metacharacter à if you want to find just a period, you have to escape it with two backslashes in R
    - “MAC.” is different than “MAC\\.”
  + *character sets*: use brackets to define sets of characters to match (ex: “MAC[B-Z]” will find MAC followed by any capital letter other than A)
  + *alternation*: basically an ‘or’ statement (ex: “MAC(B|D)” finds MAC followed by B OR D”
  + *anchors*: use ^ to anchor a pattern to the beginning or $ to anchor it to the end
  + *repetitions*: can specify the number of times you want a specific pattern to occur
    - ? is zero or one time, \* is zero or more times, and + is one or more times
* *Analyzing textual data*: example here uses arXiv which is a repository of scientific papers that are stored in the aRxiv R package and tries to find a crowdsource definition of data science
  + Example first examines whether there is more interest (more mentions) in data science in recent years à its been essentially exponential increase since 2007
  + *Corpus*: collection of many text documents
    - After compiling all of the documents, they stripped them of useless stuff like numbers, punctuation, white-space, filler words like ‘the’ and ‘and’
  + *Word clouds*: now that we have data, we can construct a word cloud that shows word frequencies (the larger it appears, the more times it was used)
  + *Document term matrices*: the term frequency inverse document frequency measure the prevalence of a term across a set of documents
    - This is kind of how Google identifies relevant searches à if a word appears a lot in one document compared to other documents, it will be more relevant for the search à use DocumentTermMatrix() for this and then use findFreqTerms() to find the actual words that have the highest scores
      * This gives slightly different results than a word cloud (and is probably more meaningful in most cases)
    - Can also do things like find terms that show up with in the same documents as the term ‘statistics’
* *Ingesting text*: bringing data into R and cleaning it up to be in a usable form
  + Example comes from Beatles Wikipedia page à can use text patterns to find things in messy data
    - Also did some document term matrices on their song titles
  + Scraping data from twitter: need to use twitteR and set up an API key and then you can find tweets with certain hashtags and stuff
    - Can analyze the tweets à distribution of the number of characters in the tweets, retweet counts (which is skewed right and follows a power law distribution), how many tweets are geolocated
    - You can’t take that many tweets at a time but you can run queries regularly and start building up a database
    - Can find trending topics in your area to see what’s going on
* Further resources: Project Gutenberg stores old books, google has n-gram sequences (look into this more?), Wikipedia has a bunch of helpful stuff, Tidy Text Mining in R is accessible on GitHub
* **R packages**: aRxiv is used in example in 15.2, tm is a text mining package, wordcloud R package, rvest helps to bring text data into R, twitteR, gutenbergr package has text data to play with, tidytext

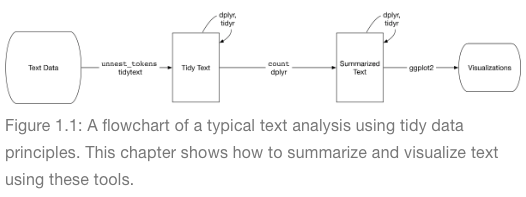
**Text Mining with R**

**Preface**

* Have a look at the tidytext package
* Outline:
  + Ch 1: unnest\_tokens(), gutenbergr and janeaustenr packages which provide data
  + Ch 2: sentiment analysis on tidy text data, uses sentiments dataset and inner\_join()
  + Ch 3: tf-idf statistic - used to identify terms that are important to a particular document
  + Ch 4: n-grams, how to analyze word networks using widyr and ggraph packages
  + Ch 5: Shows how to tidy document-term matrices and corpus objects from the ™ and quanteda packages
  + Ch 6: topic modeling, tidy() method for interpreting and visualizing the output of the topicmodels package
  + Ch 7: Case study of text analysis on tweets
  + Ch 8: Case study on NASA datasets
  + Ch 9: Case study on Usenet messages from newsgroups to look at patterns
  + This book *does not* include: clustering, classification, prediction, word embedding, more complex tokenization

**Chapter One: The tidy text format**

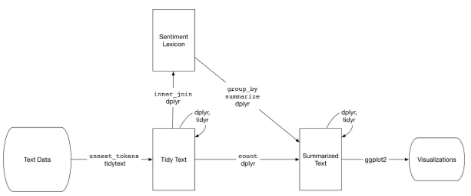
* Tidy text data is a table with one token per row (where a token is a meaningful unit of text)
* *Tokenization* is the process of splitting text into tokens
  + Can be a single word, n-gram, sentence, or paragraph
* 1.1 Contrasting tidy text with other data structures
  + Common forms of storing text: strings (usually how it comes in), corpus (raw strings annotated with additional metadata and details), document-term matrix (sparse, each row represents a document and there is a column for each term)
* 1.2 The unnest\_tokens function
  + Take some text, convert it into a data frame
  + unnest\_tokens() will split the text into individual tokens and make it tidy
    - This automatically takes away punctuation and makes everything lowercase which both make things easier to combine later on



* 1.3 Tidying the works of Jane Austen
  + Janeaustenr package - provides text in one row per line format
  + Can remove stop words by loading stop\_words dataset and then using anti\_join to get them out of your data
    - Then you can do some more analysis such as finding most common words and visualizing that info using ggplot
* 1.4 The gutenbergr package
  + Public access to the Project Gutenberg collection - can basically download a bunch of different books that are semi tidy to begin with, which is nice
* 1.4 Word frequencies
  + Comparing Jane Austen, HG Wells, and Bronte sister works in terms of word frequencies by using some tidyr functions
  + Did some graphing of actual words to show how similar they were between authors
  + Did a correlation test between the writings - think about doing this
* 1.6 Summary
  + If you get your data into the correct format, you can do quite a bit on text data just with the tidyverse

**Chapter Two: Sentiment analysis with tidy data**

* Emotional intent of words → can infer whether text is positive or negative at the least and then you can extract more meaningful emotions as well

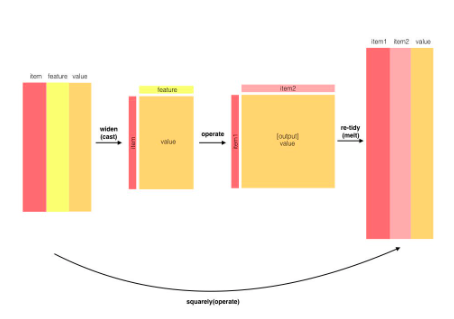


* 2.1 The sentiments dataset
  + In the tidytext package - it has three separate lexicons that are all a bit different but essentially convey the same ideas
  + These were validated by some combo of crowdsourcing reviews and data from twitter, restaurants, and movies → so maybe applying them to other things (like classic novels) may be slightly less accurate but it still works
  + There are some more specific lexicons for areas such as finance
  + Since these only account for one word, ‘no good’ would be classified as positive rather than negative
* 2.2 Sentiment analysis with inner join
  + All you really have to do is join a tidy dataset onto the sentiment dataset you’re using → then you can count things like the most common words in a book that are associated with joy
  + Can do things like calculate how sentiment changed over the course of a novel by using smaller chunks of lines (80 lines used here)
  + Can make nice plots showing the changes in sentiments here
* 2.3 Comparing the three sentiment dictionaries
  + They did sentiment analysis using three different dictionaries on the same book and compared the results
    - They had similar results in terms of when things were deemed positive and negative but the absolute value of how positive/negative it was differed
  + NRC sentiment tends to be high, AFINN has more variance, and Bing has longer stretches of similar text
* 2.4 Most common positive and negative words
  + Be sure to look at your results and make sure that the words actually make sense
    - Like ‘miss’ in Jane Austen is not a negative sentiment, it's how she is addressing women
* 2.5 Wordclouds
  + Wordcloud package → do the anti\_join() with the stop words, count the words, make the wordcloud
  + Can separate into positive and negative word clouds by using acast() and comparison.cloud()
* 2.6 Looking at units beyond just words
  + Using packages like coreNLP, cleanNLP, and sentimentr to look at things like entire sentences and trying to parse whether they are positive or negative
  + Make sure to look at how it splits up sentences (especially with dialogue) because we may want to do some other manipulating before we split them up
  + Can look at things like which chapter has the highest proportion of negative sentiments

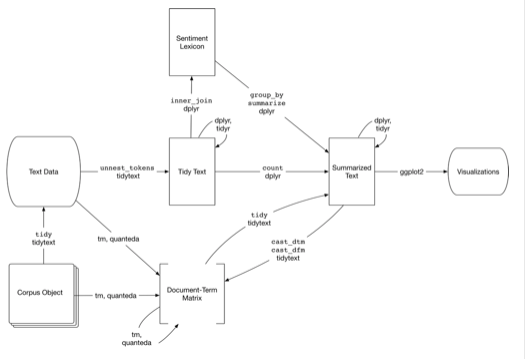
**Chapter Three: Analyzing word and document frequency: tf-idf**

* Term frequency - how frequently a word occurs in a document (be careful or accidentally counting stop words)
* Inverse document frequency - decreases the weight for common words and increases the weight for words that aren’t used much in a collection of documents (i.e. it can identify documents relevant to your search)
* 3.1 Term frequency in Jane Austen’s novels
  + Visualized the term frequency in the books (number of occurrences of that term over the total words in that book)
* 3.2 Zipf’s law
  + The right skewed distributions seen in 3.1 are common in a lot of texts
  + The frequency that a word appears is inversely proportional to its rank
  + Visualize: plot rank on the x and term frequency on the y (on log scales)
  + Can look at power law things here - the example overall wasn’t quite straight but subsections of it were
  + Interpret the results of the plot as an exercise → what does it mean to be under or over the line for both rare and common words?
* 3.3 The bind\_tf\_idf function
  + The corpus here is going to be all of Jane Austen’s novels → going to calculate the words that are important to a text but not too common
  + In this case, the words with the highest tf-idf scores were names of characters
  + Whole point: identify words that are important to one document within a collection of documents
* 3.4 A corpus of physics texts
  + Really just a case study where they run through the process again
  + Another lesson in actually looking at the results and seeing if they make sense
  + Are all of the results real words? Why are there sometimes weird ones?
* 3.5 Summary
  + Term frequency and inverse document frequency are nice → it can show of language is used in natural language (tidytext package used here)

**Chapter Four: Relationships between words: n-grams and correlations**

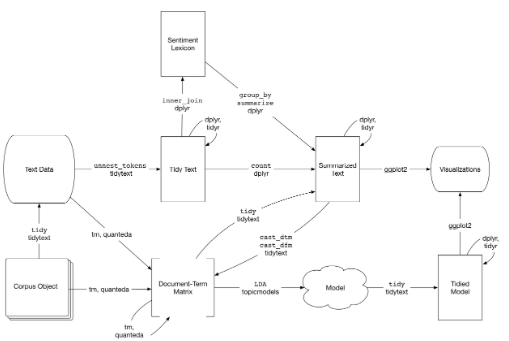
* Can look at whether words tend to follow certain other words
* Do certain words often co occur in the same documents
* Using token=”ngrams” argument as well as ggraph and widyr packages
* 4.1 Tokenizing by n-gram
  + N-grams are consecutive sequences of words → can build models describing relationships between words
  + Can split tokens into single words and remove stop words (still an n-gram though even though it may have words erased from doing this)
  + Can do things like search based on one of the words (ex: what words precede the word ‘street’?)
  + Can look at tf-idf for bigrams too → in Jane Austen examples, these are character names such as ‘captain wentworth’
  + Bigrams are good for larger text datasets → may want to stick to one word for smaller ones
  + Using bigrams to provide context in sentiment analysis → i.e. this allows you to catch phrases like ‘not happy’ as negative instead of just seeing ‘happy’ and assigning it to be positive
    - Can visualize which wrongly identified sentiments are impacting the text the most with the AFINN scoring system (multiply score by number of occurrences)
  + Visualizing a network of bigrams with ggraph → add nodes, edges, and text to graphs to visualize networks of words
    - Can also add shading to show how common a link is, arrows to show the direction of the relationships
  + Can write functions to do tidying faster/easier
* 4.2 Counting and correlating pairs of words with the widyr package
  + The widyr package counts and correlates pairs of values in a tidy dataset (can look at chapters in a book, a collection of documents, a designated number of lines, etc)
    - Pairwise\_count function will result in one row for each pair of words in the word variable → can count words co-occurring within sections
    - Different types of tidy data structures result here → just be cautious of form you have and if that is the right form to actually do what you want
  + You can examine correlation among words → use the phi coefficient here for binary outcomes
    - Can specify words and find their corresponding most highly correlated words
    - Again, you can make network-ish graphs out of these

Chapter Five: Converting to and from non-tidy formats

* Most existing R packages (other than tidytext) are not compatible with what we know to be ‘tidy data’ (tidyr, ggplot, dplyr)
* 5.1 Tidying a document-term matrix
  + Each row is a document (book, article, etc), each column is a term, each value is the number of appearances of that term in that document
  + These are usually sparse since most terms don’t appear in most documents
  + Can use tidy() to turn a document-term matrix into a tidy data frame in the broom package
  + Can use cast() to turn a tidy dataset into a matrix (tidytext can do this) → 3 forms
    - Watch which document form matrix you’re using → they can come from different packages and use different tidying verbs
  + Can see how words change in frequency over time by extracting the year (or another time unit)
* 5.2 Casting tidy text data into a matrix
  + Basically shows different ways to change tidy data into a matrix (can be sparse or not) → decisions to be made here
  + Want the matrix for machine learning applications mostly
* 5.3 Tidying corpus objects with metadata
  + Metadata is basically the data *about* the data
  + Oftentimes want to keep metadata attached → tidy() does this well
  + [Tm.plugin.webmining](https://cran.r-project.org/package=tm.plugin.webmining) → can find news articles based on some keyword
    - Go through examples to see how data taken from here is cleaned/used
    - Financial articles → looking at stocks, exploring common words, could attempt to predict changes in the market, be careful of words here AGAIN
  + New lexicon for using financial data → can see which words are most commonly used in articles grouped by their sentiment
  + There is a ‘positivity score’ that shows how positive an article or company or what have you is compared to others

Chapter Six: Topic Modeling

* Unsupervised classification of documents into clusters → finds groups that we weren’t sure we were looking for



* Latent Dirichlet allocation (LDA) is a popular method for fitting a topic model
  + Documents can overlap with each other in terms of content rather than being separated into discrete groups
  + LDA comes from the topicmodels R package
* 6.1 Latent Dirichlet allocation
  + Every document is a mixture of topics → can designate how many topics you want
  + Every topic is a mixture of words
  + This algorithm basically tries to find the best balance of these things
  + Word-topic probabilities: calculates the likelihood of a certain word being part of a certain topic → can find words with highest probabilities, can see which words are the most different between the topics; can help to see differences between the topics
  + Document-topic probabilities: can model each document as a mixture of topics - basically estimates the proportion of words in a document that are from a specific topic
* 6.2 Example: the great library heist
  + Took four things that they knew to be different and tested to see if the algorithm could sort them properly → can do this on unlabeled data (you don’t know what group it belongs to)
  + This example split into books and chapters (each unit is a chapter of a book) and then into tokens → had to split into a matrix to use the topicmodels package
  + Looked a per document per topic probabilities to see if they could get the chapters in the right books in the right order
  + Did a good job with three of the books and an alright job with the fourth
  + Can use augment() to find which words in each document were assigned to each topic
  + Can make confusion matrices to show how often the consensus matches with reality → here it is how often words from one book were assigned to another
  + Often the mistakes LDA makes are reasonable but also sometimes you get some that don’t make much sense
* 6.3 Alternative LDA implementations
  + There are other packages that take slightly different approaches
  + Mallet package is one example of this

**Chapter Seven: Case Study in Comparing Twitter Archives**

* Several lexicons were designed to be used on Tweets
* 7.1 Getting the data and distribution of tweets
  + Used twitter directions to get all of their tweets and did some univariate graphs just to get a feel for the data
* 7.2 Word frequencies
  + Cleaned the data (removed retweets and links and characters like ‘&’), removed stop words, created tokens, and found word frequencies (count of *that* word divided by total count of words)
  + A good example of using spread() from tidyr to change a data frame for easy plotting (also examples of good ggplot things that could be used)
* 7.3 Comparing word usage
  + Compared log odds ratios to see which words were more likely to come from one user or the other
    - Plotted the top 15 most distinctive words from each of their accounts
* 7.4 Changes in word use
  + How have the words they use in their tweets changed over time?
  + They chose to compare from month to month → just created a month variable
  + They also used nest() to create miniature data frames for each word
  + They used glm with the binomial option to model the miniature data frames and then extracted the ‘important’ ones (note: adjusted p-values for MCs)
    - Visualized these with line graphs that showed the trends
* 7.5 Favorites and retweets
  + Got their RT and favorite info from the Twitter API
  + Broke down data to see the median number of retweets for words
  + Also did the same to see which words led to more favorites

**Chapter Eight: Case Study in mining NASA metadata**

* Overview is to use the metadata in NASA data to connect some of their different types of datasets
* Can they find datasets that are related? Are there groups of similar datasets?
* 8.1 How data is organized at NASA
  + All open access and can be gotten in a JSON file → gong to focus on title, description, and keywords for each dataset
  + Use unnest to get info out of lists
  + Initial exploring: what are the most common words in the dataset titles, keywords and descriptions? (keep in mind that you can make your own stop words and remove them)
* 8.2 Word co-occurrences and correlations
  + Use pairwise\_count() from widyr package to count how many times each pair of words occurs together in a title or description
  + Made cool network graphs → check to see if there actually are clusters of if it would be better to look at things like tf-idf to find important relationships
  + Find correlations between keywords → can get numbers AND network graphs from correlations
    - This one offers more information because it shows which keywords occur more often *together* than with other keywords whereas the first networks just kind of told us which keywords pairs occur most often
* 8.3 Calculating tf-idf for the description fields
  + Found the highest tf-idfs for descriptions but the results were all from one word descriptions and maybe these should be thrown out (it deems the one word *very* important since there's only one)
  + Got the description words with tf-idf and then found the highest tf-idf words for a given keyword
    - Identify important description words for each keyword
* 8.4 Topic modeling
  + Remember: this models each document as a mixture of topics and each topic as a mixture of words
  + Created a document term matrix → first need to create a count of how many times each word is used in each document
  + The actual modeling: they tried different numbers of clusters and settled on 24, which created nice topics without having worrisome probabilities for being assigned to a topic
  + Looked at top ten words in each topic to try and discern what the topics were about → they end up being vaguely like the description fields
  + Looked at probability that a document belonged to a certain topic
    - Look at values close to zero and one → is it doing a good job of sorting documents into topics?
  + Found keywords associated with each topic → did a deeper look and this can have future implications → can help humans assign future keywords based on descriptions

**Chapter Nine: Case Study in analyzing usenet text**

* Analysis of 20,000 messages sent to 20 usenet boards in 1993 (<http://qwone.com/~jason/20Newsgroups/>)
* 9.1 Pre-processing
  + Reading in this data is a bit tricky → read files from folders into a data frame, then unnest() and map() to apply reading in the folders to further subfolders
  + Looking at which newsgroups were included and how many messages were posted in each of them
  + Did lots of good cleaning here → removed useless stuff like headers and signatures (LOOK AT YOUR DATA)
* 9.2 Words in newsgroups
  + Look at word counts overall or broken down by newsgroup
  + Found tf-idf within newsgroups
  + What newsgroups tend to be similar to each other in text content?
    - Use correlations to explore a question like this
    - More visualizing networks (this made four distinct groups)
  + Did some topic modeling with four different science related boards
    - Wanted to see if the algorithm would come up with the same topics
* 9.3 Sentiment analysis
  + A general look at which boards were the most positive and negative
  + Closer look at which words have the largest impact (positive or negative)
  + Calculated each word’s contribution to each newsgroup’s sentiment score to find the strongest contributors
  + Found the most positive/negative individual messages → good check to see if your analysis is working as you had thought
  + N-gram analysis → looked for bigrams that may have opposite meaning than what the sentiment analysis determined

**Humanities Data in R**

**Chapter Nine: Natural Language Processing**

* Natural language processing pipeline → text data requires a lot of pre processing to get it into a form that you can actually work with
* 9.2 Tokenization and Sentence Splitting
  + Tokenization: the process of splitting text into meaningful elements
  + Stanford CoreNLP (coreNLP is the corresponding R package)
* 9.3 Lemmatization and Part of Speech Tagging
  + A lemma kind of represents a group of words → ‘go’ is a lemma for ‘gone’, ‘going’, and ‘went’; ‘mice’ becomes ‘mouse’
  + Universal tag-set → smaller group of parts of speech to determine lemmas and also just look at writing structure
* 9.4 Dependencies
  + Sentence parsing: where the words within a sentence are assigned a linguistic structure that links all of the parts together
  + Dependencies give the relationships between pairs of lemmas which can then create an entire ‘parse tree’
* 9.5 Named Entity Recognition
  + This is the task of automatically detecting and classifying elements of a text into broad semantic categories
    - Can find things like location, dates, times, people (characters)
* 9.6 Coreference
  + Quantifying the relative importance of words in a text → in their example they set out to find which character is referred to the most (it was Sherlock Holmes)
  + Identifies that things like ‘Sheila went to the store and she bought a tomato’ are referring to the same person

**Chapter Ten: Text Analysis**

* 10.4 Stylometric Analysis
  + Characterizes writing style → can find patterns in a particular author’s writing over time as well as can be used for identifying authorship
  + Look at things like speech bigrams → how often is a verb followed immediately by a noun?
    - They do this with principal component analysis → *if this becomes something that I’ll use, then I’ll learn it*

**Data Mining and Business Analysis with R**

**Chapter 19: Text Mining and Sentiment Analysis**

* Stem words: cut words into their root → ‘tax’ comes from ‘taxing’, ‘taxes’, and ‘taxation’ → Porter stemming algorithm
* Removing stop words *and* words with low frequencies (very rare words)
* Used inverse multinomial logistic model to predict what rating a restaurant would have based on bigrams in the reviews → use mnlm in textir package

**Beginning Data Science with R**

**Chapter Eight: Text Mining**

* 8.2 Dataset
  + 2000 movies(1000 positive, 1000 negative) → which words are ‘most positive’ and can you guess whether individual movies will have ‘positive’ or ‘negative’ sentiment
* 8.3 Reading Text Input Data
  + Plain text is the easiest format to read text → doesn’t include a lot of metadata
  + An example on how to get the data and create a corpus using the ™ package
* 8.4 Common Text Preprocessing Tasks
  + Remove punctuation and stop words and stem the document
  + Remove stop words that are unique to what you are doing → with these movie reviews, they removed ‘film’ and ‘movie’ since they are relatively meaningless here
* 8.5 Term Document Matrix
  + Bag of words → gives the text some structure as it is stored → need this for most text mining algorithms
  + Rows are unique words in the corpus, columns are the documents (can go the opposite way though and that is called a document term matrix)
* 8.6 Text Mining Applications
  + Frequency Analysis: counting the number of words in a corpus, doing word associations
  + Text classification: is a document part of a group? Is it or is not spam? Is it a positive review or a negative review?
  + Train and test datasets → train model and then test it out. How well does it do? How do you measure that? Do you have problems with dimensionality here? Do you have to make choices about what terms to train the model on to save memory and computation time?
  + *Support vector machines → look into this*