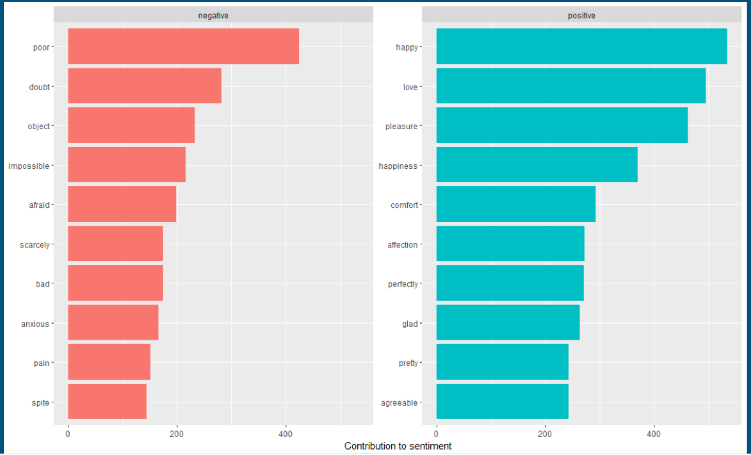
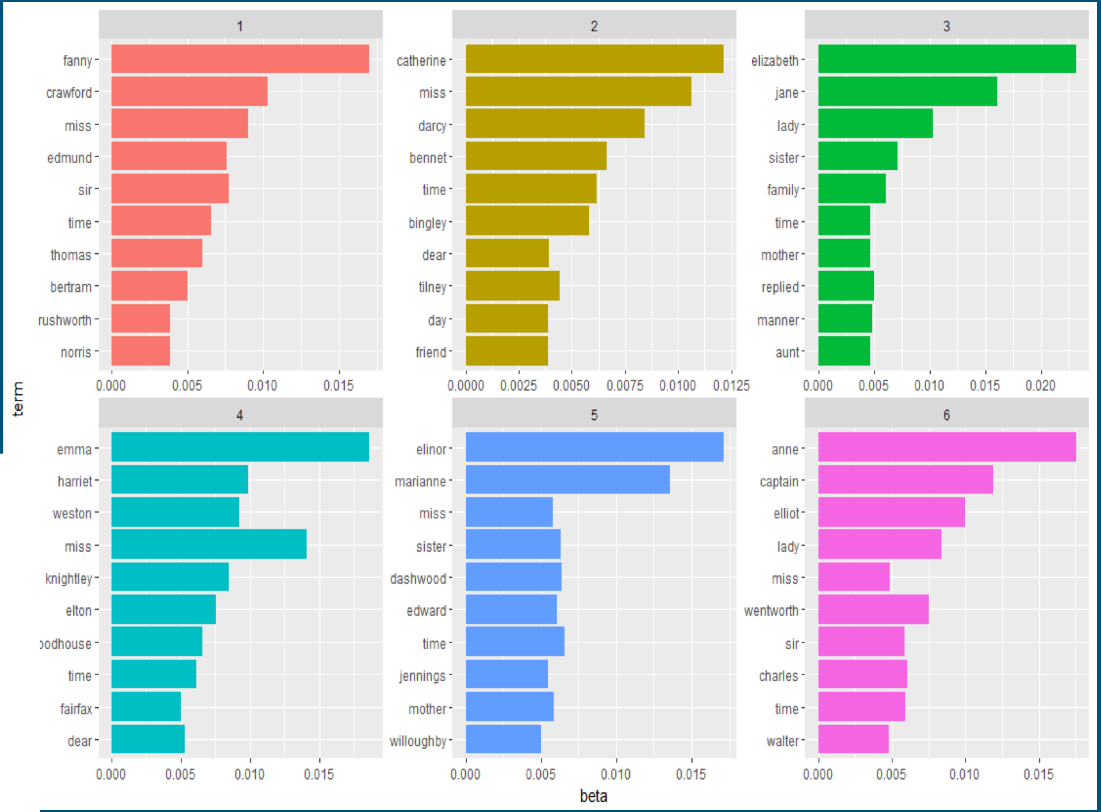
Text Mining with R

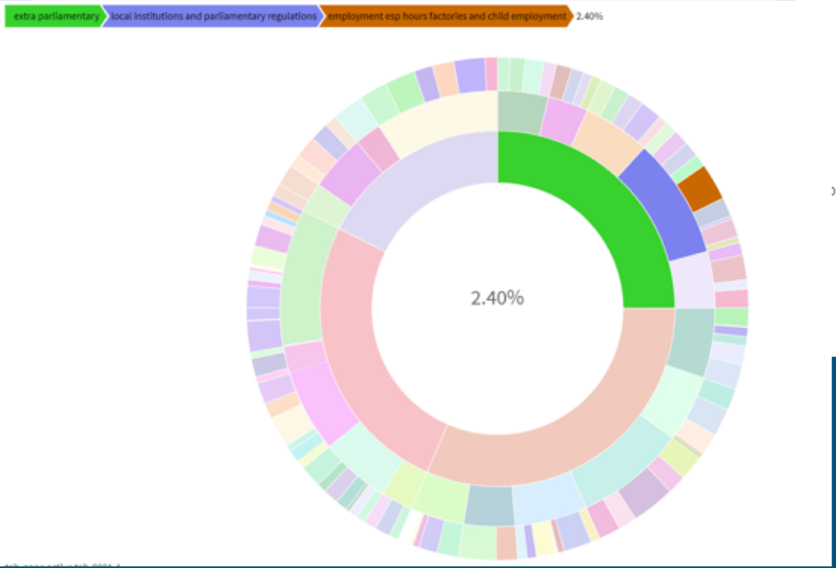
Ben Williams
2018



negative
positive
poor
doubt
object
impossible
afraid
scarcely
bad
anxious
pain
spite
happy
love
pleasure
happiness
comfort
affection
perfectly
glad
pretty
agreeable
0
200
400
Contribution to sentiment



1
2
3
4
5
6
term
beta
fanny
crawford
miss
edmund
sir
time
thomas
bertram
rushworth
norris
catherine
miss
darcy
bennet
time
bingley
dear
tilney
day
friend
elizabeth
jane
lady
sister
family
time
mother
replied
manner
aunt
emma
harriet
weston
miss
knightley
elton
oodhouse
time
fairfax
dear
elinor
marianne
miss
sister
dashwood
edward
time
jennings
mother
willoughby
anne
captain
elliot
lady
miss
wentworth
sir
charles
time
walter
0.000
0.005
0.010
0.015
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.020



extra parliamentary
local institutions and parliamentary regulations
employment esp hours factories and child employment
2.40%
2.40%

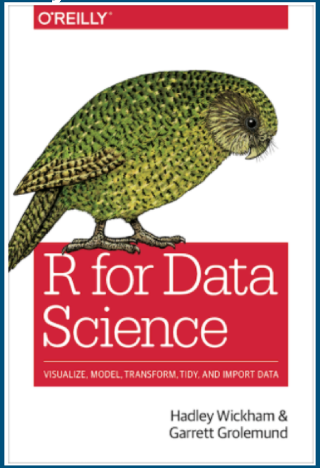
Resources

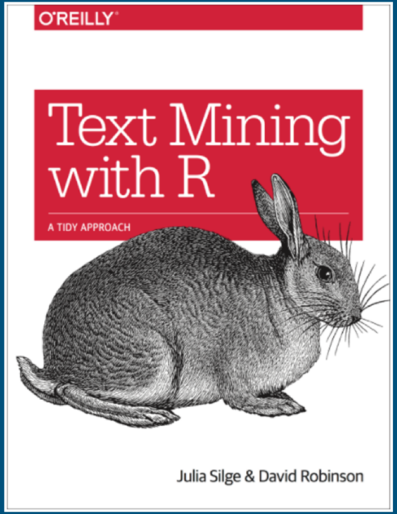
Text Mining with R: Julia Silge (StackOverflow) & David Robinson (DataCamp)

https://www.tidytextmining.com/

R for Data Science: Garrett Grolemund & Hadley Wickham

http://r4ds.had.co.nz/


O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham & Garrett Grolemund



Both are free!

Tidy Data

In general:

-> 1 observation per row, 1 variable per column

Text Mining:

-> One token per row

Token: word, bigram, n-gram, etc.

Tools for Tidying Data

tidyverse packages: *dplyr, ggplot2, tidyr, stringr, readr* (the package *tidyverse* contains many of the useful packages and loads them all at once)

group_by()/ungroup(): group by a variable, then perform groupwise operations

filter() : filter rows

select(): select columns

count(): count the number of observations in a group

mutate(): add a new column

%>% : “composed of”, “then”

Brief Aside if Necessary

%>% is called a *pipe* (see R for Data Science 5.6 for more info)

The %>% lets us easily and clearly combine functions in R

x %>% f(y) really means, f(x,y).

Example: data_stat_club is dataset of everyone’s name, age, birthplace

```
data_stat_club %>%
      select(age) %>%
      mean(na.rm=T)
```
#this takes the tibble data_stat_club, selects the variable *age*, and gets its mean

Data

Can read .csv, .tsv, .xlsx, etc. into R. Look up *readr* and *readxl* package for more info. i.e. read_csv()

We want data formatted as a *data frame* or as a *tibble* (a data frame that prints to the console nicely)

Want: Text in one column of the tibble, does not have to be one token-per-row to be read into R

Tidy Text Data

Package: *tidytext*

Function: unnest_tokens(tbl,output,input,token=”words”)

unnest_tokens() takes your data (tibble or data frame) and a given character column and *tokenizes* that column. By default, it splits the column into words

This is the first step in tidying the data.

See first part of R Code

Stop Words

Stop Words are words we assume are uninformative in any sort of textual analysis, such as “the”, “and”, “is”, “were”, etc.

tidytext has provided a tibble of stop words called *stop_words*. The columns are “word” and “lexicon”

We can remove stop words from our newly tidy text data using anti_join()

```
text_data %>% #unclean data is text_data
    unnest_tokens(text,word) %>% #input column is “text”, output is “word”
    anti_join(stop_words) #remove any stop words
```

Sentiment Analysis

Idea: sometimes a word has an emotion or sentiment associated with it. We can analyze the text based on these emotions

For example: “joyful” might be classified as positive, and “distraught” might be classified as negative

Somewhat ad hoc in my mind: i.e. “not happy” -> “happy” without stop words -> classified as positive. There had been work done on negating words though...

Lexicons built into *tidytext* package, can also specialize it for your own text

Topic Modeling

Latent Dirichlet Allocation (LDA) Topic Modeling is an unsupervised algorithm that “groups” a corpus into a given number of topics.

In LDA each document is represented by a distribution of topics which are characterized by a distribution over the unique words in the corpus (Blei, Ng and Jordan, 2003)

Think of Dallas Morning News, say we model it with 4 topics.
1: (president, mayor, vote, county, judge, senate,...)
2: (golf, hockey, Dirk, cowboys, basketball, soccer, …)
3: (sunny, rain, sun, wind, cold, flood, temperature, high,...)
4: (police, crime, prison, bail, officer, shooting, robbery,...)
Each newspaper article is made up of these topics, each topic is a distribution over all the unique words in the corpus of newspapers

Document Term Matrix (DTM)

Matrix where rows are documents of a corpus, and columns are terms in vocabulary

A DTM is the input into an LDA model, along with the parameter for number of topics

Transform tidy data to DTM: cast_dtm(data,document,term,count)

Tidy a DTM: tidy(dtm)

Beta and Gamma

Beta: per-topic-per-word probability

-Use to see what words are important in each topic

Gamma: per-document-per-topic probability

-Use to see what topics make up each document

Shiny Tool - if time

https://github.com/williamsbenjamin/nesting-topics

app_comp.R and app_hand.R are Shiny scripts that make a Sunburst of hierarchically nested topic models. They use two datasets available on my github. Check out the datasets to see the format for creating a Sunburst. Really easy and a great interactive tool! Sunburst is a D3 visualization that has been transferred to an R package as well.

Questions?

benjamin@smu.edu