Text Mining with R

Text Mining with R - A Tidy Approach Julia Silge and David Robinson <https://www.tidytextmining.com/>

# Chapter 7: Case Study: Comparing Twitter Archives

One type of text that gets plenty of attention is text shared online via Twitter. In fact, several of the sentiment lexicons used in this book (and commonly used in general) were designed for use with and validated on tweets. Both of the authors of this book are on Twitter and are fairly regular users of it, so in this case study, let’s compare the entire Twitter archives of Julia and David.

## 7.1 Getting the data and distribution of tweets

An individual can download their own Twitter archive by following directions available on Twitter’s website. <https://help.twitter.com/ja/managing-your-account/how-to-download-your-twitter-archive>

We each downloaded ours and will now open them up. Let’s use the lubridate package to convert the string timestamps to date-time objects and initially take a look at our tweeting patterns overall (Figure 7.1).

julia and dave tweet data: <https://github.com/kojimizu/tidy-text-mining/tree/master/data>

# package load  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.5.1

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.1

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:lubridate':  
##   
## intersect, setdiff, union

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(readr)  
  
tweets\_julia <- read\_csv("C:/Users/kojikm.mizumura/Desktop/Data Science/Text Mining with R/II. Case Study/Data/tweets\_julia.csv")

## Parsed with column specification:  
## cols(  
## tweet\_id = col\_double(),  
## in\_reply\_to\_status\_id = col\_double(),  
## in\_reply\_to\_user\_id = col\_double(),  
## timestamp = col\_character(),  
## source = col\_character(),  
## text = col\_character(),  
## retweeted\_status\_id = col\_double(),  
## retweeted\_status\_user\_id = col\_double(),  
## retweeted\_status\_timestamp = col\_character(),  
## expanded\_urls = col\_character()  
## )

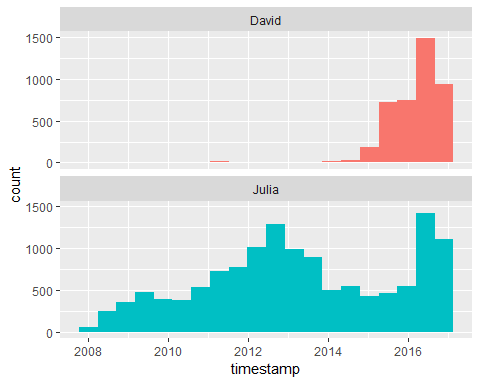
tweets\_dave <- read\_csv("C:/Users/kojikm.mizumura/Desktop/Data Science/Text Mining with R/II. Case Study/Data/tweets\_dave.csv")

## Parsed with column specification:  
## cols(  
## tweet\_id = col\_double(),  
## in\_reply\_to\_status\_id = col\_double(),  
## in\_reply\_to\_user\_id = col\_double(),  
## timestamp = col\_character(),  
## source = col\_character(),  
## text = col\_character(),  
## retweeted\_status\_id = col\_double(),  
## retweeted\_status\_user\_id = col\_double(),  
## retweeted\_status\_timestamp = col\_character(),  
## expanded\_urls = col\_character()  
## )

head(tweets\_dave)

## # A tibble: 6 x 10  
## tweet\_id in\_reply\_to\_stat~ in\_reply\_to\_use~ timestamp source text   
## <dbl> <dbl> <dbl> <chr> <chr> <chr>   
## 1 8.16e17 NA NA 2017-01-~ "<a hre~ RT @Park~  
## 2 8.16e17 NA NA 2017-01-~ "<a hre~ RT @Caus~  
## 3 8.16e17 NA NA 2017-01-~ "<a hre~ RT @hadl~  
## 4 8.16e17 NA NA 2017-01-~ "<a hre~ "RT @the~  
## 5 8.15e17 NA NA 2017-01-~ "<a hre~ RT @elli~  
## 6 8.15e17 8.15e17 3230388598 2016-12-~ "<a hre~ "@dataan~  
## # ... with 4 more variables: retweeted\_status\_id <dbl>,  
## # retweeted\_status\_user\_id <dbl>, retweeted\_status\_timestamp <chr>,  
## # expanded\_urls <chr>

tweets <- bind\_rows(tweets\_julia %>%   
 mutate(person = "Julia"),  
 tweets\_dave %>%   
 mutate(person = "David")) %>%  
 mutate(timestamp = ymd\_hms(timestamp))  
  
ggplot(tweets, aes(x = timestamp, fill = person)) +  
 geom\_histogram(position = "identity", bins = 20, show.legend = FALSE) +  
 facet\_wrap(~person, ncol = 1)



David and Julia tweet at about the same rate currently and joined Twitter about a year apart from each other, but there were about 5 years where David was not active on Twitter and Julia was. In total, Julia has about 4 times as many tweets as David.

## 7.2 Word frequencies

Let’s use unnest\_tokens() to make a tidy data frame of all the words in our tweets, and remove the common English stop words. There are certain conventions in how people use text on Twitter, so we will do a bit more owrk with our text here than, for example, we did with the narrative text from Project Gutenberg.

First, we will remove tweets from this dataset that are retweets so that we only have tweets that we wrote ourselves. Next, the mutate() line removes links and cleans our some characters that we don’t want like ampersands and such.

In the call to unnest\_tokens(), we unnest using a regex pattern, instead of just looking for single unigrams (words). This regex pattern is very useful for dealing with Twitter text; it retains hashtags and mentions of usernames with the @ symbol.

Because we have kept these types of symbols in the texzt, we can’t use a simple anti\_join() to remove stop words. Instead, we can take the approach shown in the filter() line that uses str\_detect() from the stringr package.

library(tidytext)

## Warning: package 'tidytext' was built under R version 3.5.1

library(stringr)  
  
replace\_reg <- "https://t.co/[A-Za-z\\d]+|http://[A-Za-z\\d]+|&amp;|&lt;|&gt;|RT|https"  
unnest\_reg <- "([^A-Za-z\_\\d#@']|'(?![A-Za-z\_\\d#@]))"  
  
tweets

## # A tibble: 17,265 x 11  
## tweet\_id in\_reply\_to\_stat~ in\_reply\_to\_use~ timestamp source   
## <dbl> <dbl> <dbl> <dttm> <chr>   
## 1 8.16e17 8.16e17 33559167 2017-01-01 21:48:41 "<a hr~  
## 2 8.16e17 8.16e17 13074042 2017-01-01 21:16:16 "<a hr~  
## 3 8.16e17 8.16e17 13074042 2017-01-01 21:13:45 "<a hr~  
## 4 8.16e17 NA NA 2017-01-01 21:12:15 "<a hr~  
## 5 8.16e17 8.15e17 14173097 2017-01-01 14:21:48 "<a hr~  
## 6 8.15e17 8.15e17 1823987821 2017-01-01 07:08:41 "<a hr~  
## 7 8.15e17 8.15e17 2837127738 2017-01-01 07:08:19 "<a hr~  
## 8 8.15e17 8.15e17 1823987821 2017-01-01 07:06:50 "<a hr~  
## 9 8.15e17 8.15e17 13074042 2017-01-01 07:02:34 "<a hr~  
## 10 8.15e17 8.15e17 13074042 2017-01-01 05:18:59 "<a hr~  
## # ... with 17,255 more rows, and 6 more variables: text <chr>,  
## # retweeted\_status\_id <dbl>, retweeted\_status\_user\_id <dbl>,  
## # retweeted\_status\_timestamp <chr>, expanded\_urls <chr>, person <chr>

# need to find out this formula for tidy data cleaning   
tidy\_tweets <- tweets %>%   
 filter(!str\_detect(text, "^RT")) %>%  
 mutate(text = str\_replace\_all(text, replace\_reg, "")) %>%  
 unnest\_tokens(word, text, token = "regex", pattern = unnest\_reg) %>%  
 filter(!word %in% stop\_words$word,  
 str\_detect(word, "[a-z]"))  
  
tidy\_tweets %>% select(tweet\_id, timestamp,person,word)

## # A tibble: 94,733 x 4  
## tweet\_id timestamp person word   
## <dbl> <dttm> <chr> <chr>   
## 1 678288082 2008-02-05 00:00:00 Julia ron   
## 2 678288082 2008-02-05 00:00:00 Julia paul   
## 3 678288082 2008-02-05 00:00:00 Julia called   
## 4 678288082 2008-02-05 00:00:00 Julia house   
## 5 678288082 2008-02-05 00:00:00 Julia vote   
## 6 678288082 2008-02-05 00:00:00 Julia tomorrow  
## 7 678689892 2008-02-05 00:00:00 Julia browse   
## 8 678689892 2008-02-05 00:00:00 Julia burda   
## 9 678689892 2008-02-05 00:00:00 Julia sewing   
## 10 678689892 2008-02-05 00:00:00 Julia magazine  
## # ... with 94,723 more rows

Now we can calculate word frequencies for each person. First, we group by person and count how many times each person used each word. Then we use left\_join() to add a column of the total number of words used by each person. (This is higher for Julia than David since she has more tweets than David.) Finally, we calculate a frequency for each person and word.

frequency <- tidy\_tweets %>%   
 group\_by(person) %>%   
 count(word,sort=TRUE) %>%   
 left\_join(tidy\_tweets %>%   
 group\_by(person) %>%   
 summarise(total=n())) %>%   
 mutate(freq=n/total)

## Joining, by = "person"

frequency

## # A tibble: 20,736 x 5  
## # Groups: person [2]  
## person word n total freq  
## <chr> <chr> <int> <int> <dbl>  
## 1 Julia time 584 74572 0.00783  
## 2 Julia @selkie1970 570 74572 0.00764  
## 3 Julia @skedman 531 74572 0.00712  
## 4 Julia day 467 74572 0.00626  
## 5 Julia baby 408 74572 0.00547  
## 6 David @hadleywickham 315 20161 0.0156   
## 7 Julia love 304 74572 0.00408  
## 8 Julia @haleynburke 299 74572 0.00401  
## 9 Julia house 289 74572 0.00388  
## 10 Julia morning 278 74572 0.00373  
## # ... with 20,726 more rows

This is a nice and tidy data frame but we would actually like to plot those frequencies on the x- and y-axes of a plot, so we will need to use spread() from tidyr make a differently shaped data frame.

library(tidyr)  
  
frequency <- frequency %>%   
 select(person,word,freq) %>%   
 spread(person,freq) %>%   
 arrange(Julia,David) %>%   
  
frequency

Now this is ready for us to plot. Let’s use geom\_jitter() so that we don’t see the discreteness at the low end of frequency as much, and check\_overlap = TRUE so the text labels don’t all print out on top of each other (only some will print).

library(scales)

##   
## Attaching package: 'scales'

## The following object is masked from 'package:readr':  
##   
## col\_factor

library(ggplot2)  
  
frequency

## [1] 1

# ggplot(frequency, aes(Julia, David)) +  
# geom\_jitter(alpha = 0.1, size = 2.5, width = 0.25, height = 0.25) +  
# geom\_text(aes(label = word), check\_overlap = TRUE, vjust = 1.5) +  
# scale\_x\_log10(labels = percent\_format()) +  
# scale\_y\_log10(labels = percent\_format()) +  
# geom\_abline(color = "red")

Words near the line in Figure 7.2 are used with about equal frequencies by David and Julia, while words far away from the line are used much more by one person compared to the other. Words, hashtags, and usernames that appear in this plot are ones that we have both used at least once in tweets.

This may not even need to be pointed out, but David and Julia have used their Twitter accounts rather differently over the course of the past several years. David has used his Twitter account almost exclusively for professional purposes since he became more active, while Julia used it for entirely personal purposes until late 2015 and still uses it more personally than David. We see these differences immediately in this plot exploring word frequencies, and they will continue to be obvious in the rest of this chapter.

## 7.3 Comparing word usage

We just made a plot comparing raw word frequencies over our whole Twitter histories; now let’s find which words are more or less likely to come from each person’s account using **the log odds** ratio. First, let’s restrict the analysis moving forward to tweets from David and Julia sent during 2016. David was consistently active on Twitter for all of 2016 and this was about when Julia transitioned into data science as a career.

tidy\_tweets <- tidy\_tweets %>%   
 filter(timestamp >= as.Date("2016-01-01"),  
 timestamp <=as.Date("2017-01-01"))  
  
tidy\_tweets%>% select(tweet\_id, timestamp,person,word)

## # A tibble: 25,908 x 4  
## tweet\_id timestamp person word   
## <dbl> <dttm> <chr> <chr>   
## 1 6.83e17 2016-01-01 21:50:14 Julia blue   
## 2 6.83e17 2016-01-01 21:50:14 Julia castle   
## 3 6.83e17 2016-01-01 21:50:14 Julia montgomery  
## 4 6.83e17 2016-01-01 21:50:14 Julia rating   
## 5 6.83e17 2016-01-01 21:50:14 Julia stars   
## 6 6.83e17 2016-01-01 21:50:14 Julia read   
## 7 6.83e17 2016-01-01 21:50:14 Julia suspicion   
## 8 6.83e17 2016-01-01 21:51:53 David @quominus   
## 9 6.83e17 2016-01-01 21:51:53 David im   
## 10 6.83e17 2016-01-02 06:16:15 Julia watched   
## # ... with 25,898 more rows

Next, let’s use str\_detect() to remove Twitter usernames from the word column, because otherwise, the results here are dominated only by people who Julia or David know and the other does not. After removing these, we count how many times each person uses each word and keep only the words used more than 10 times. After a spread() operation, we can calculate the log odds ratio for each word, using

where *n* is the number of times the word in question is used by each person and the total indicates the total words for each person.

library(magrittr)

##   
## Attaching package: 'magrittr'

## The following object is masked from 'package:tidyr':  
##   
## extract

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.5.1

## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --

## √ tibble 1.4.2 √ purrr 0.2.5  
## √ tibble 1.4.2 √ forcats 0.3.0

## Warning: package 'purrr' was built under R version 3.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x lubridate::as.difftime() masks base::as.difftime()  
## x scales::col\_factor() masks readr::col\_factor()  
## x lubridate::date() masks base::date()  
## x purrr::discard() masks scales::discard()  
## x magrittr::extract() masks tidyr::extract()  
## x dplyr::filter() masks stats::filter()  
## x lubridate::intersect() masks base::intersect()  
## x dplyr::lag() masks stats::lag()  
## x purrr::set\_names() masks magrittr::set\_names()  
## x lubridate::setdiff() masks base::setdiff()  
## x lubridate::union() masks base::union()

word\_ratios <- tidy\_tweets %>%   
 filter(!str\_detect(word,"^@")) %>%   
 count(word,person) %>%   
 filter(sum(n)>=10) %>%   
 ungroup() %>%   
 spread(person,n,fill=0) %>%   
 mutate\_if(is.numeric,funs((.+1)/sum(.+1))) %>%   
 mutate(logratio=log(David/Julia)) %>%   
 arrange(desc(logratio))

What are some words that have been about equally likely to come from David or Julia’s account during 2016?

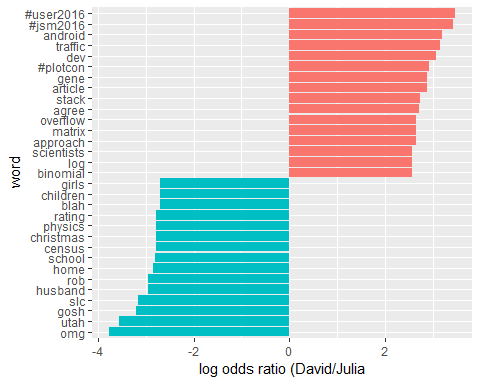
word\_ratios %>%   
 arrange(abs(logratio))

## # A tibble: 6,688 x 4  
## word David Julia logratio  
## <chr> <dbl> <dbl> <dbl>  
## 1 idea 0.00129 0.00133 -0.0245  
## 2 map 0.000619 0.000603 0.0263  
## 3 science 0.00152 0.00157 -0.0313  
## 4 email 0.000563 0.000543 0.0364  
## 5 file 0.000563 0.000543 0.0364  
## 6 names 0.00101 0.000965 0.0488  
## 7 account 0.000450 0.000422 0.0645  
## 8 api 0.000450 0.000422 0.0645  
## 9 function 0.000900 0.000844 0.0645  
## 10 population 0.000450 0.000422 0.0645  
## # ... with 6,678 more rows

We are about equally likelt to tweer about maps, email, APIs, and functions.

Which words are most likely to be from Julia’s account or from David’s account? Let’s just take the top 15 most distinctive words fro each account and plot them in Figure 7.3.

word\_ratios %>%   
 group\_by(logratio <0) %>%   
 top\_n(15,abs(logratio)) %>%   
 ungroup() %>%   
 mutate(word=reorder(word,logratio)) %>%   
 ggplot(aes(word,logratio,fill=logratio<0))+  
 geom\_col(show.legend = F)+  
 coord\_flip()+  
 ylab("log odds ratio (David/Julia")+  
 scale\_fill\_discrete(name="",labels=c("David","Julia"))



So David has tweeted about specific conferences he has gone to, genes, Stack Overflow, and matrices while Julia tweeted about Utah, physics, Census data, Christmas, and her family.

## 7.4 Changes in word use

The section above looked at overall word use, but now let’s ask a different question. Which words’ frequencies have changed the fastest in our Twitter feeds? Or to state this another way, which words have we tweeted about at a higher or lower rate as time has passed? To do this, we will define a new time variable in the data frame that defines which unit of time each tweet was posted in. We can use floor\_date() from lubridate to do this, with a unit of our choosing; using 1 month seems to work well for this year of tweets from both of us.

After we have the time bins defined, we count how many times each of us used each word in each time bin. After that, we add columns to the data frame for the total number of words used in each time bin by each person and the total number of times each word was used by each person. We can then filter() to only keep words used at least some minimum number of times (30, in this case).

library(lubridate)  
  
words\_by\_time <- tidy\_tweets %>%   
 filter(!str\_detect(word,"^@")) %>%   
 mutate(time\_floor=floor\_date(timestamp,unit="1 month")) %>%   
 count(time\_floor,person,word) %>%   
 ungroup() %>%   
 group\_by(person,time\_floor) %>%   
 mutate(time\_total=sum(n)) %>%   
 group\_by(word) %>%   
 mutate(word\_total=sum(n)) %>%   
 ungroup() %>%   
 rename(count=n) %>%   
 filter(word\_total>30)  
  
words\_by\_time

## # A tibble: 970 x 6  
## time\_floor person word count time\_total word\_total  
## <dttm> <chr> <chr> <int> <int> <int>  
## 1 2016-01-01 00:00:00 David #rstats 2 307 324  
## 2 2016-01-01 00:00:00 David bad 1 307 33  
## 3 2016-01-01 00:00:00 David bit 2 307 45  
## 4 2016-01-01 00:00:00 David blog 1 307 60  
## 5 2016-01-01 00:00:00 David broom 2 307 41  
## 6 2016-01-01 00:00:00 David call 2 307 31  
## 7 2016-01-01 00:00:00 David check 1 307 42  
## 8 2016-01-01 00:00:00 David code 3 307 49  
## 9 2016-01-01 00:00:00 David data 2 307 276  
## 10 2016-01-01 00:00:00 David day 2 307 65  
## # ... with 960 more rows

Each row in this data frame corresponds to one person using one word in a given time bin. The count column tells us how many times that person used that word in that time bin, the time\_total column tells us how many words that person used during that time bin, and the word\_total column tells us how many times that person used that word over the whole year. This is the data set we can use for modeling.

We can use nest() from tidyr to make a data frame with a list column that contains little miniature data frames for each word. Let’s do that now nad take a look at the resulting structure.

nested\_data <- words\_by\_time %>%   
 nest(-word,-person)  
nested\_data

## # A tibble: 112 x 3  
## person word data   
## <chr> <chr> <list>   
## 1 David #rstats <tibble [12 x 4]>  
## 2 David bad <tibble [9 x 4]>   
## 3 David bit <tibble [10 x 4]>  
## 4 David blog <tibble [12 x 4]>  
## 5 David broom <tibble [10 x 4]>  
## 6 David call <tibble [9 x 4]>   
## 7 David check <tibble [12 x 4]>  
## 8 David code <tibble [10 x 4]>  
## 9 David data <tibble [12 x 4]>  
## 10 David day <tibble [8 x 4]>   
## # ... with 102 more rows

## # A tibble: 112 x 3  
## person word data   
## <chr> <chr> <list>   
## 1 David #rstats <tibble [12 × 4]>  
## 2 David bad <tibble [9 × 4]>   
## 3 David bit <tibble [10 × 4]>  
## 4 David blog <tibble [12 × 4]>  
## 5 David broom <tibble [10 × 4]>  
## 6 David call <tibble [9 × 4]>   
## 7 David check <tibble [12 × 4]>  
## 8 David code <tibble [10 × 4]>  
## 9 David data <tibble [12 × 4]>  
## 10 David day <tibble [8 × 4]>   
## # ... with 102 more rows

This data frame has one row for each person-word combination; the data column is a list column that contains data frames, one for each combination of person and word. Let’s use map() from the **purrr** library to apply our modeling procedure to each of those little data frames inside our big data frame. This is count data so let’s use glm() with family = "binomial" for modeling.

library(purrr)  
  
nested\_models <- nested\_data %>%   
 mutate(models=map(data,~glm(cbind(count,time\_total)~  
 time\_floor,.,family="binomial")))  
nested\_models

## # A tibble: 112 x 4  
## person word data models   
## <chr> <chr> <list> <list>   
## 1 David #rstats <tibble [12 x 4]> <S3: glm>  
## 2 David bad <tibble [9 x 4]> <S3: glm>  
## 3 David bit <tibble [10 x 4]> <S3: glm>  
## 4 David blog <tibble [12 x 4]> <S3: glm>  
## 5 David broom <tibble [10 x 4]> <S3: glm>  
## 6 David call <tibble [9 x 4]> <S3: glm>  
## 7 David check <tibble [12 x 4]> <S3: glm>  
## 8 David code <tibble [10 x 4]> <S3: glm>  
## 9 David data <tibble [12 x 4]> <S3: glm>  
## 10 David day <tibble [8 x 4]> <S3: glm>  
## # ... with 102 more rows

Now notice that we have a new column for the modeling results; it is another list column and contains glm objects. The next step is to use map() and tidy() from the **broom** package to pull out the slopes for each of these models and find the important ones. We are comparing many slopes here and some of them are not statistically significant, so let’s apply an adjustment to the p-values for multiple comparisons.

library(broom)

## Warning: package 'broom' was built under R version 3.5.1

library(tidytext)  
  
slopes <- nested\_models %>%   
 unnest(map(models,tidy)) %>%   
 filter(term=="time\_floor") %>%   
 mutate(adjusted.p.value=p.adjust(p.value))

Now let’s find the most important slopes. Which words have changed in frequency at a moderately significant level in our tweets?

slopes

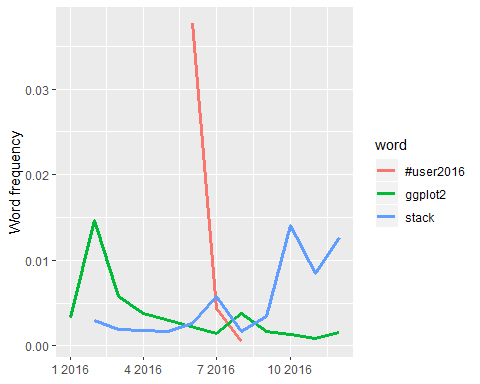
## # A tibble: 111 x 8  
## person word term estimate std.error statistic p.value  
## <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 David #rstats time\_floor -4.95e-9 0.00000000828 -0.598 0.550   
## 2 David bad time\_floor -2.92e-8 0.0000000291 -1.00 0.316   
## 3 David bit time\_floor 1.47e-8 0.0000000275 0.534 0.593   
## 4 David blog time\_floor -2.97e-8 0.0000000229 -1.30 0.194   
## 5 David broom time\_floor -2.04e-8 0.0000000187 -1.09 0.276   
## 6 David call time\_floor -4.71e-8 0.0000000290 -1.63 0.104   
## 7 David check time\_floor -1.06e-8 0.0000000236 -0.446 0.655   
## 8 David code time\_floor -9.51e-9 0.0000000219 -0.433 0.665   
## 9 David data time\_floor 1.99e-8 0.00000000948 2.10 0.0359  
## 10 David day time\_floor -1.39e-8 0.0000000259 -0.538 0.591   
## # ... with 101 more rows, and 1 more variable: adjusted.p.value <dbl>

top\_slopes <- slopes %>%   
 filter(adjusted.p.value<0.1)  
top\_slopes

## # A tibble: 6 x 8  
## person word term estimate std.error statistic p.value adjusted.p.value  
## <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 David ggpl~ time~ -8.26e-8 1.97e-8 -4.20 2.72e-5 0.00300   
## 2 Julia #rst~ time~ -4.50e-8 1.12e-8 -4.02 5.93e-5 0.00647   
## 3 Julia post time~ -4.82e-8 1.45e-8 -3.31 9.23e-4 0.0978   
## 4 Julia read time~ -9.33e-8 2.54e-8 -3.67 2.44e-4 0.0263   
## 5 David stack time~ 8.04e-8 2.19e-8 3.67 2.46e-4 0.0263   
## 6 David #use~ time~ -8.18e-7 1.55e-7 -5.27 1.33e-7 0.0000148

To visualize our results, we can plot these words’ use for both David and Julia over this year of tweets.

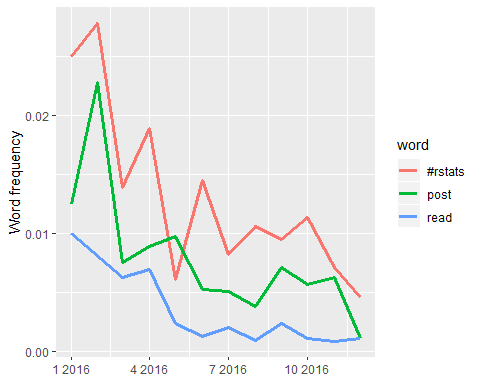
words\_by\_time %>%   
 inner\_join(top\_slopes,by=c("word","person")) %>%   
 filter(person=="David") %>%   
 ggplot(aes(time\_floor,count/time\_total,color=word))+  
 geom\_line(size=1.3)+  
 labs(x=NULL,y="Word frequency")



We see in Figure 7.4 that David tweeted a lot about the UseR conference while he was there and then quickly stopped. He has tweeted more about Stack Overflow toward the end of the year and less about ggplot2 as the year has progressed.

Now let’s plot words that have changed frequency in Julia’s tweets in Figure 7.5.

words\_by\_time %>%   
 inner\_join(top\_slopes,by=c("word","person")) %>%   
 filter(person=="Julia") %>%   
 ggplot(aes(time\_floor,count/time\_total,color=word))+  
 geom\_line(size=1.3)+  
 labs(x=NULL,y="Word frequency")



All the significant slopes for Julia are negative. This means she has not tweeted at a higher rate using any specific words, but instead using a variety of different words; her tweets earlier in the year contained the words shown in this plot at higher proportions. Words she uses when publicizing a new blog post like the #rstats hashtag and “post” have gone down in frequency, and she has tweeted less about reading.

## 7.5 Favorites and retweets

Another important characteristic of tweets is how many times they are favorited or retweeted. Let’s explore which words are more likely to be retweeted or favorited for Julia’s and David’s tweets. When a user downloads their own Twitter archive, favorites and retweets are not included, so we constructed another dataset of the authors’ tweets that includes this information. We accessed our own tweets via the Twitter API and downloaded about 3200 tweets for each person. In both cases, that is about the last 18 months worth of Twitter activity. This corresponds to a period of increasing activity and increasing numbers of followers for both of us.

library(tidyverse)  
library(tidytext)  
library(lubridate)  
  
# data load  
tweets\_julia <- read\_csv("C:/Users/kojikm.mizumura/Desktop/Data Science/Text Mining with R/II. Case Study/Data/juliasilge\_tweets.csv")

## Parsed with column specification:  
## cols(  
## id = col\_double(),  
## created\_at = col\_datetime(format = ""),  
## source = col\_character(),  
## retweets = col\_integer(),  
## favorites = col\_integer(),  
## text = col\_character()  
## )

tweets\_dave <- read\_csv("C:/Users/kojikm.mizumura/Desktop/Data Science/Text Mining with R/II. Case Study/Data/drob\_tweets.csv")

## Parsed with column specification:  
## cols(  
## id = col\_double(),  
## created\_at = col\_datetime(format = ""),  
## source = col\_character(),  
## retweets = col\_integer(),  
## favorites = col\_integer(),  
## text = col\_character()  
## )

# data merge  
tweets <- bind\_rows(tweets\_julia %>%   
 mutate(person="Julia"),  
 tweets\_dave %>%   
 mutate(person="David")) %>%   
 mutate(created\_at=ymd\_hms(created\_at))

Now that we have this second, smaller set of only recent tweets, let’s use unnest\_tokens() to transform these tweets to a tidy data set. Let’s remove all retweets and replies from this data set so we only look at regular tweets that David and Julia have posted directly.

tidy\_tweets <- tweets %>%   
 filter(!str\_detect(text, "^(RT|@)")) %>%  
 mutate(text = str\_replace\_all(text, replace\_reg, "")) %>%  
 unnest\_tokens(word, text, token = "regex", pattern = unnest\_reg) %>%  
 anti\_join(stop\_words)

## Joining, by = "word"

tidy\_tweets

## # A tibble: 11,078 x 7  
## id created\_at source retweets favorites person word   
## <dbl> <dttm> <chr> <int> <int> <chr> <chr>   
## 1 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia violet  
## 2 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia hubble  
## 3 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia space   
## 4 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia teles~  
## 5 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia kinde~  
## 6 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia week   
## 7 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia pretty  
## 8 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia happy   
## 9 5.94e17 2015-05-01 02:49:02 Twitter f~ 0 0 Julia life   
## 10 5.94e17 2015-05-01 02:49:02 Twitter f~ 0 0 Julia pee   
## # ... with 11,068 more rows

To start with, let’s look at the number of times each of our tweets was retweeted. Let’s find the total number of retweets for each person.

tidy\_tweets

## # A tibble: 11,078 x 7  
## id created\_at source retweets favorites person word   
## <dbl> <dttm> <chr> <int> <int> <chr> <chr>   
## 1 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia violet  
## 2 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia hubble  
## 3 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia space   
## 4 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia teles~  
## 5 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia kinde~  
## 6 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia week   
## 7 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia pretty  
## 8 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia happy   
## 9 5.94e17 2015-05-01 02:49:02 Twitter f~ 0 0 Julia life   
## 10 5.94e17 2015-05-01 02:49:02 Twitter f~ 0 0 Julia pee   
## # ... with 11,068 more rows

totals <- tidy\_tweets %>%   
 group\_by(person,id) %>%   
 summarise(rts=sum(retweets)) %>%   
 group\_by(person) %>%   
 summarise(total\_rts=sum(rts))  
totals

## # A tibble: 2 x 2  
## person total\_rts  
## <chr> <int>  
## 1 David 110171  
## 2 Julia 12701

Now let’s find the median number of retweets for each word and person. We probably want to count each tweet/word combination only once, so we will use group\_by() and summarise() twice, one right after the other. The first summarise() statement counts how many times each word was retweeted, for each tweet and person. In the second summarise() statement, we can find the median retweets for each person and word, also count the number of times each word was used ever by each person and keep that in uses. Next, we can join this to the data frame of retweet totals. Let’s filter() to only keep words mentioned at least 5 times.

library(magrittr)  
library(tidyverse)  
  
tidy\_tweets

## # A tibble: 11,078 x 7  
## id created\_at source retweets favorites person word   
## <dbl> <dttm> <chr> <int> <int> <chr> <chr>   
## 1 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia violet  
## 2 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia hubble  
## 3 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia space   
## 4 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia teles~  
## 5 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia kinde~  
## 6 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia week   
## 7 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia pretty  
## 8 5.94e17 2015-05-01 01:57:38 Instagram 0 0 Julia happy   
## 9 5.94e17 2015-05-01 02:49:02 Twitter f~ 0 0 Julia life   
## 10 5.94e17 2015-05-01 02:49:02 Twitter f~ 0 0 Julia pee   
## # ... with 11,068 more rows

word\_by\_rts <- tidy\_tweets %>%   
 group\_by(id, word, person) %>%   
 summarise(rts = first(retweets)) %>%   
 group\_by(person, word) %>%   
 summarise(retweets = median(rts), uses = n()) %>%  
 left\_join(totals) %>%  
 filter(retweets != 0) %>%  
 ungroup()

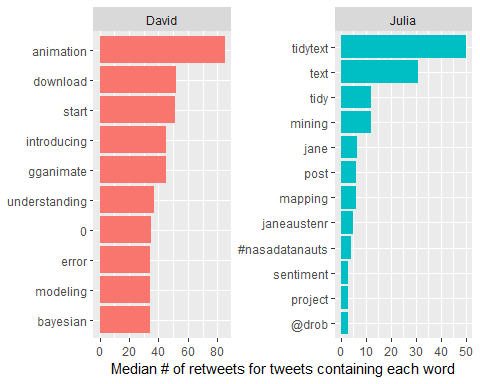
## Joining, by = "person"

word\_by\_rts %>%   
 filter(uses >= 5) %>%  
 arrange(desc(retweets))

## # A tibble: 178 x 5  
## person word retweets uses total\_rts  
## <chr> <chr> <dbl> <int> <int>  
## 1 David animation 85 5 110171  
## 2 David download 52 5 110171  
## 3 David start 51 7 110171  
## 4 Julia tidytext 50 7 12701  
## 5 David gganimate 45 8 110171  
## 6 David introducing 45 6 110171  
## 7 David understanding 37 6 110171  
## 8 David 0 35 7 110171  
## 9 David error 34.5 8 110171  
## 10 David bayesian 34 7 110171  
## # ... with 168 more rows

At the top of this sorted data frame, we see tweets from Julia and David about packages that they work on, like gutenbergr, gganimate, and tidytext. Let’s plot the words that have the highest median retweets for each of our accounts (Figure 7.6)

word\_by\_rts %>%  
 filter(uses >= 5) %>%  
 group\_by(person) %>%  
 top\_n(10, retweets) %>%  
 arrange(retweets) %>%  
 ungroup() %>%  
 mutate(word = factor(word, unique(word))) %>%  
 ungroup() %>%  
 ggplot(aes(word, retweets, fill = person)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~ person, scales = "free", ncol = 2) +  
 coord\_flip() +  
 labs(x = NULL,   
 y = "Median # of retweets for tweets containing each word")



We see lots of word about R packages, including tidytext, a package about which you are reading right now! The “0” for David comes from tweets where he mentions version numbers of packages, like “broom 0.4.0” or similar.

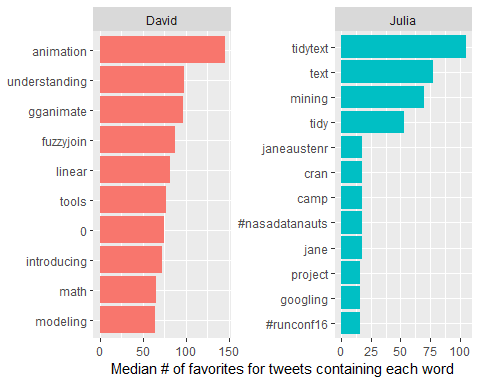
We can follow a similar procedure to see which words led to more favorites. Are they different than the words that lead to more retweets?

totals <- tidy\_tweets %>%   
 group\_by(person, id) %>%   
 summarise(favs = sum(favorites)) %>%   
 group\_by(person) %>%   
 summarise(total\_favs = sum(favs))  
  
word\_by\_favs <- tidy\_tweets %>%   
 group\_by(id, word, person) %>%   
 summarise(favs = first(favorites)) %>%   
 group\_by(person, word) %>%   
 summarise(favorites = median(favs), uses = n()) %>%  
 left\_join(totals) %>%  
 filter(favorites != 0) %>%  
 ungroup()

## Joining, by = "person"

We have bult the data frames we need. Now let’s make our visualization in the following figure.

word\_by\_favs %>%  
 filter(uses >= 5) %>%  
 group\_by(person) %>%  
 top\_n(10, favorites) %>%  
 arrange(favorites) %>%  
 ungroup() %>%  
 mutate(word = factor(word, unique(word))) %>%  
 ungroup() %>%  
 ggplot(aes(word, favorites, fill = person)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~ person, scales = "free", ncol = 2) +  
 coord\_flip() +  
 labs(x = NULL,   
 y = "Median # of favorites for tweets containing each word")



We see some minor differences between Figures 7.6 and 7.7, especially near the bottom of the top 10 list, but these are largely the same words as for retweets. In general, the same words that lead to retweets lead to favorites. A prominent word for Julia in both plots is the hashtag for the NASA Datanauts program that she has participated in; read on to Chapter 8 to learn more about NASA data and what we can learn from text analysis of NASA datasets.

## 7.6 Summary

This chapter was our first case study, a beginning-to-end analysis that demonstrates how to bring together the concepts and code we have been exploring in a cohesive way to understand a text data set. Comparing word frequencies allows us to see which words we tweeted more and less frequently, and the log odds ratio shows us which words are more likely to be tweeted from each of our accounts. We can use nest() and map() with the glm() function to find which words we have tweeted at higher and lower rates as time has passed. Finally, we can find which words in our tweets led to higher numbers of retweets and favorites. All of these are examples of approaches to measure how we use words in similar and different ways and how the characteristics of our tweets are changing or compare with each other. These are flexible approaches to text mining that can be applied to other types of text as well.

# Chaoter 8: Case Study mining NASA metadata

There are over 32,000 datasets hosted and/or maintained by NASA; these datasets cover topics from Earth science to aerospace engineering to management of NASA itself. We can use the metadata for these datasets to understand the connections between them. <https://www.nasa.gov/>

The metadata includes information like the title of the dataset, a description filed, what organization(s) within NASA is responsible for the dataset, keywords for the dataset that have been assigned by a human being, and so forth.

NASA places a high priority on making its data open and accessible, even requiring all NASA-funded research to be openly accessible online. The metadata for all its datasets is publicly available online in JSON format. <https://www.nasa.gov/press-release/nasa-unveils-new-public-web-portal-for-research-results> <https://data.nasa.gov/data.json>

In this chapter, we will treat the NASA metadata as a text dataset and show how to implement several tidy text approaches with this real-life text. We will use word co-occurrences and correlations, tf-idf, and topic modeling to explore the connections between the datasets. Can we find datasets that are related to each other? Can we find clusters of similar datasets? Since we have several text fields in the NASA metadata, most importantly the title, description, and keyword fields, we can explore the connections between the fields to better understand the complex world of data at NASA. This type of approach can be extended to any domain that deals with text, so let’s take a look at this metadata and get started.

txt\_jp <- "羽鳥来日にあわせて猛者が次々に集結したという感じ"  
txt\_jp

## [1] "羽鳥来日にあわせて猛者が次々に集結したという感じ"

quanteda::tokens(txt\_jp)

## tokens from 1 document.  
## text1 :  
## [1] "羽鳥" "来日" "に" "あわせて" "猛者" "が"   
## [7] "次々に" "集結" "した" "という" "感じ"

## 8.1 How data is organized at NASA

First, let’s download the JASON file and take a look at the names of what is stored in the metadata.

# install.packages("jsonlite")  
library(jsonlite)

## Warning: package 'jsonlite' was built under R version 3.5.1

##   
## Attaching package: 'jsonlite'

## The following object is masked from 'package:purrr':  
##   
## flatten

# Json file download does not work with fromJson() function  
# metadata <- fromJSON("https://data.nasa.gov/data.jason")  
# https://github.com/nasa/data.nasa.gov/tree/master/js  
  
metadata <- fromJSON("C:/Users/kojikm.mizumura/Desktop/Data Science/Text Mining with R/data1.json")  
  
names(metadata$dataset)

## [1] "@type" "accessLevel" "accrualPeriodicity"  
## [4] "bureauCode" "contactPoint" "description"   
## [7] "distribution" "identifier" "issued"   
## [10] "keyword" "landingPage" "language"   
## [13] "modified" "programCode" "publisher"   
## [16] "title" "license" "\_id"   
## [19] "spatial" "temporal" "theme"   
## [22] "references" "rights" "describedBy"

We see here that we could extract information from who publishes each dataset to what license they are released under.

It seems likely that the title, description, and keywords for each dataset may be most fruitful for drawing connections between datasets. Let’s check them out.

class(metadata$dataset$title)

## [1] "character"

class(metadata$dataset$description)

## [1] "character"

class(metadata$dataset$keyword)

## [1] "list"

The title and description fields are stored as character vectors, but the keywords are stored as a list of character vectors.

### 8.1.1 Wrangling and tidying the data

Let’s set up separate tidy data frames for title, description, and keyword, keeping the dataset ids fro each so that we can connect them later in the analaysis if necessary.

library(dplyr)  
  
nasa\_title <- data\_frame(id=metadata$dataset$`\_id`$`$oid`,title=metadata$dataset$title)  
nasa\_title

## # A tibble: 42,966 x 2  
## id title   
## <chr> <chr>   
## 1 <NA> Global Landslide Catalog Export   
## 2 <NA> The NASA Air Traffic Management Ontology (atm~  
## 3 55942a57c63a7fe59b495a78 15 Minute Stream Flow Data: USGS (FIFE)   
## 4 55942a58c63a7fe59b495a79 15 Minute Stream Flow Data: USGS (FIFE)   
## 5 55942a58c63a7fe59b495a7b 2000 Pilot Environmental Sustainability Index~  
## 6 55942a58c63a7fe59b495a7c 2000 Pilot Environmental Sustainability Index~  
## 7 55942a58c63a7fe59b495a7e 2001 Environmental Sustainability Index (ESI)   
## 8 55942a58c63a7fe59b495a7f 2001 Environmental Sustainability Index (ESI)   
## 9 55942a58c63a7fe59b495a80 2001 Environmental Sustainability Index (ESI)   
## 10 55942a58c63a7fe59b495a82 2002 Environmental Sustainability Index (ESI)   
## # ... with 42,956 more rows

These are just a few example titles from the dataset we will be exploring. Notice that we have the NASA-assigned ids here, and also that there are duplicate titles on separate datasets.

nasa\_desc <- data\_frame(id=metadata$dataset$`\_id`$`$oid`,  
 desc=metadata$dataset$description)  
  
nasa\_desc %>%   
 select(desc) %>%   
 sample\_n(5)

## # A tibble: 5 x 1  
## desc   
## <chr>   
## 1 This dataset is part of the collection of Special Sensor Microwave/Imag~  
## 2 An Innovative Method of NOX Reduction Through Fuel Additives for the UE~  
## 3 "MODIS (or Moderate Resolution Imaging Spectroradiometer) is a key inst~  
## 4 "This data set consists of a southern Africa subset of the Global Histo~  
## 5 "MODIS (or Moderate Resolution Imaging Spectroradiometer) is a key inst~

Here we see the first part of several selected description fields from the metadata.

Now we can build the tidy data frame for the keywords. For this one, we need to use unnest() from tidyr, because they are in a list-column.

library(tidyr)  
nasa\_keyword <- data\_frame(id=metadata$dataset$`\_id`$`$oid`,  
 keyword=metadata$dataset$keyword) %>%   
 unnest(keyword)  
nasa\_keyword

## # A tibble: 180,626 x 2  
## id keyword   
## <chr> <chr>   
## 1 <NA> landslide   
## 2 <NA> hazards   
## 3 <NA> mudslide   
## 4 <NA> earth   
## 5 <NA> citizen science   
## 6 <NA> airspace   
## 7 <NA> aerospace   
## 8 <NA> ontology   
## 9 <NA> atm   
## 10 <NA> air traffic management  
## # ... with 180,616 more rows

This is a tidy data frame because we have one row each keyword; this means we will have multiple rows for each dataset bevcause a dataset can have more than one keyword.

Now it is a time to use tidytext’s unnest\_tokens() for the title and description fileds so we can do the text analysis. Let’s also remove stop words from the titles and descriptions. We will not remove stop words from the keywords, because those are short, human-assigned keywords like “RADIATION” or “CLIMATE INDICATORS”