Text Mining with R

KM

2018/08/06

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(tidyverse)

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

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

## √ ggplot2 3.0.0 √ purrr 0.2.5  
## √ tibble 1.4.2 √ dplyr 0.7.6  
## √ tidyr 0.8.1 √ stringr 1.3.1  
## √ readr 1.1.1 √ forcats 0.3.0

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

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

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

## -- Conflicts ------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(lubridate)

##   
## Attaching package: 'lubridate'

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

library(ggplot2)  
library(dplyr)  
library(readr)  
  
tweets\_julia <- read\_csv("U:/2. Data Science/2. Text Books/3. 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()  
## )

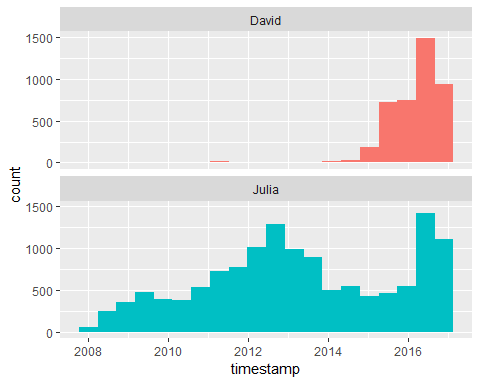
tweets\_dave <- read\_csv("U:/2. Data Science/2. Text Books/3. Text Mining with R/II. Case Study/Data/tweets\_dave.csv")

## 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\_sta~ in\_reply\_to\_use~ timestamp source text   
## <dbl> <dbl> <dbl> <chr> <chr> <chr>  
## 1 8.16e17 NA NA 2017-01-~ "<a h~ RT @~  
## 2 8.16e17 NA NA 2017-01-~ "<a h~ RT @~  
## 3 8.16e17 NA NA 2017-01-~ "<a h~ RT @~  
## 4 8.16e17 NA NA 2017-01-~ "<a h~ "RT ~  
## 5 8.15e17 NA NA 2017-01-~ "<a h~ RT @~  
## 6 8.15e17 8.15e17 3230388598 2016-12-~ "<a h~ "@da~  
## # ... 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\_sta~ in\_reply\_to\_use~ timestamp source  
## <dbl> <dbl> <dbl> <dttm> <chr>   
## 1 8.16e17 8.16e17 33559167 2017-01-01 21:48:41 "<a h~  
## 2 8.16e17 8.16e17 13074042 2017-01-01 21:16:16 "<a h~  
## 3 8.16e17 8.16e17 13074042 2017-01-01 21:13:45 "<a h~  
## 4 8.16e17 NA NA 2017-01-01 21:12:15 "<a h~  
## 5 8.16e17 8.15e17 14173097 2017-01-01 14:21:48 "<a h~  
## 6 8.15e17 8.15e17 1823987821 2017-01-01 07:08:41 "<a h~  
## 7 8.15e17 8.15e17 2837127738 2017-01-01 07:08:19 "<a h~  
## 8 8.15e17 8.15e17 1823987821 2017-01-01 07:06:50 "<a h~  
## 9 8.15e17 8.15e17 13074042 2017-01-01 07:02:34 "<a h~  
## 10 8.15e17 8.15e17 13074042 2017-01-01 05:18:59 "<a h~  
## # ... 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:purrr':  
##   
## discard

## 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:purrr':  
##   
## set\_names

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

library(tidyverse)  
  
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?

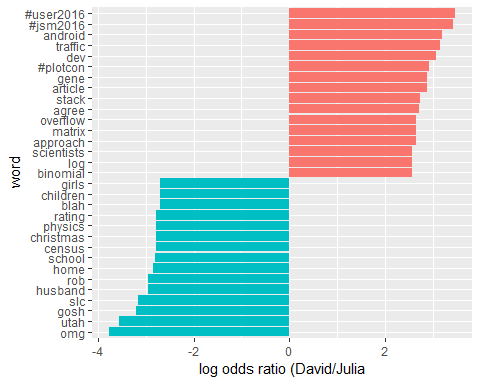
word\_ratios %>%   
 arrange(abs(logratio))

## # 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

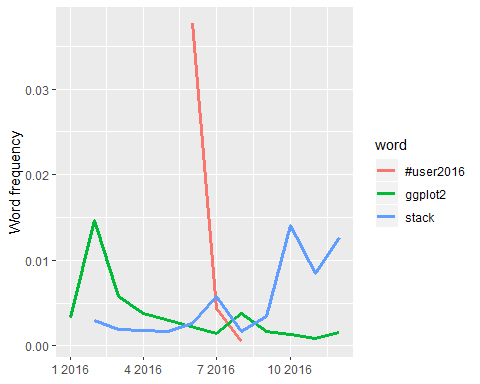
## # A tibble: 111 x 8  
## person word term estimate std.err~ statistic p.value adjusted.p.value  
## <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 David #rst~ time~ -4.95e-9 8.28e-9 -0.598 0.550 1  
## 2 David bad time~ -2.92e-8 2.91e-8 -1.00 0.316 1  
## 3 David bit time~ 1.47e-8 2.75e-8 0.534 0.593 1  
## 4 David blog time~ -2.97e-8 2.29e-8 -1.30 0.194 1  
## 5 David broom time~ -2.04e-8 1.87e-8 -1.09 0.276 1  
## 6 David call time~ -4.71e-8 2.90e-8 -1.63 0.104 1  
## 7 David check time~ -1.06e-8 2.36e-8 -0.446 0.655 1  
## 8 David code time~ -9.51e-9 2.19e-8 -0.433 0.665 1  
## 9 David data time~ 1.99e-8 9.48e-9 2.10 0.0359 1  
## 10 David day time~ -1.39e-8 2.59e-8 -0.538 0.591 1  
## # ... with 101 more rows

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.

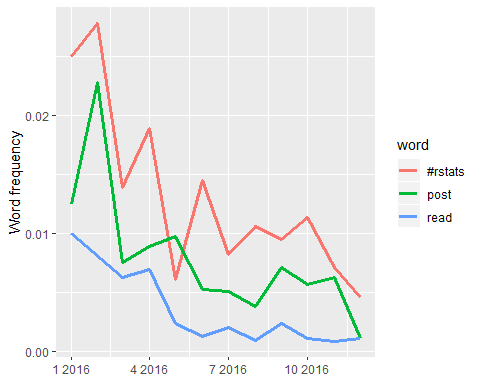
words\_by\_time %>%   
 inner\_join(top\_slopes,by=c("word","person")) %>%   
 filter(person=="David") %>%   
 ggplot(aes(time\_floor,count/time\_total,color=word))+  
 geom\_line(size=1.3)+  
 labs(x=NULL,y="Word frequency")



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("U:/2. Data Science/2. Text Books/3. 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("U:/2. Data Science/2. Text Books/3. 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()

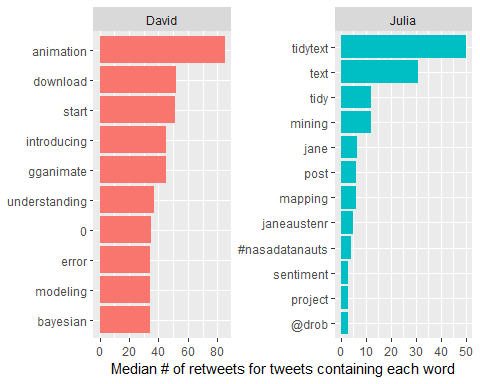
## Joining, by = "person"

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.

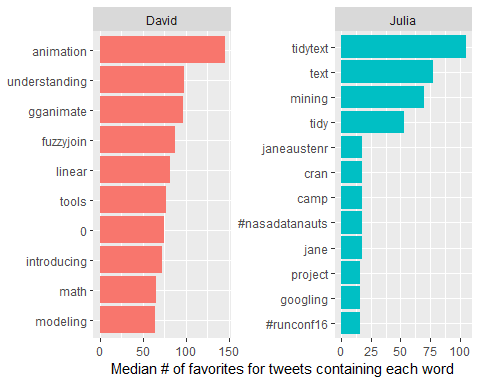
We can follow a similar procedure to see which words led to more favorites. Are they different than the words that lead to more retweets?

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.

## 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("U:/2. Data Science/2. Text Books/3. Text Mining with R/II. Case Study/Data/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 (atmo~  
## 3 55942a57c63a7fe59b495a~ 15 Minute Stream Flow Data: USGS (FIFE)   
## 4 55942a58c63a7fe59b495a~ 15 Minute Stream Flow Data: USGS (FIFE)   
## 5 55942a58c63a7fe59b495a~ 2000 Pilot Environmental Sustainability Index ~  
## 6 55942a58c63a7fe59b495a~ 2000 Pilot Environmental Sustainability Index ~  
## 7 55942a58c63a7fe59b495a~ 2001 Environmental Sustainability Index (ESI)   
## 8 55942a58c63a7fe59b495a~ 2001 Environmental Sustainability Index (ESI)   
## 9 55942a58c63a7fe59b495a~ 2001 Environmental Sustainability Index (ESI)   
## 10 55942a58c63a7fe59b495a~ 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

## # A tibble: 42,966 x 2  
## id desc   
## <chr> <chr>   
## 1 <NA> The Global Landslide Catalog (GLC) was developed w~  
## 2 <NA> The NASA ATM (Air Traffic Management) Ontology des~  
## 3 55942a57c63a7fe59b~ USGS 15 minute stream flow data for Kings Creek on~  
## 4 55942a58c63a7fe59b~ ABSTRACT: USGS 15 minute stream flow data for King~  
## 5 55942a58c63a7fe59b~ The 2000 Pilot Environmental Sustainability Index ~  
## 6 55942a58c63a7fe59b~ The 2000 Pilot Environmental Sustainability Index ~  
## 7 55942a58c63a7fe59b~ The 2001 Environmental Sustainability Index (ESI) ~  
## 8 55942a58c63a7fe59b~ The 2001 Environmental Sustainability Index (ESI) ~  
## 9 55942a58c63a7fe59b~ The 2001 Environmental Sustainability Index (ESI) ~  
## 10 55942a58c63a7fe59b~ The 2002 Environmental Sustainability Index (ESI) ~  
## # ... with 42,956 more rows

nasa\_desc %>%   
 select(desc) %>%   
 sample\_n(5)

## # A tibble: 5 x 1  
## desc   
## <chr>   
## 1 CALIPSO Lidar Level 2 1 km cloud layer data   
## 2 "The MOD09CMG Version 6 product provides an estimate of the surface spec~  
## 3 The Top One Percent Wild Areas Dataset of the Last of the Wild Project, ~  
## 4 New High-Barrier Polymer Nanocomposite Food Packaging Enables 5-Year She~  
## 5 "The SeaWiFS instrument was launched by Orbital Sciences Corporation on ~

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”

library(tidytext)  
  
  
nasa\_title <- nasa\_title %>%   
 unnest\_tokens(word, title) %>%   
 anti\_join(stop\_words)

## Joining, by = "word"

nasa\_desc <- nasa\_desc %>%   
 unnest\_tokens(word, desc) %>%   
 anti\_join(stop\_words)

## Joining, by = "word"

These are now in the tidy text format that we have been working with throughout this book, with one token (word, in this case) per row; let’s take a look before we move on in our analysis.

nasa\_title

## # A tibble: 303,466 x 2  
## id word   
## <chr> <chr>   
## 1 <NA> global   
## 2 <NA> landslide   
## 3 <NA> catalog   
## 4 <NA> export   
## 5 <NA> nasa   
## 6 <NA> air   
## 7 <NA> traffic   
## 8 <NA> management  
## 9 <NA> ontology   
## 10 <NA> atmonto   
## # ... with 303,456 more rows

nasa\_desc

## # A tibble: 3,734,573 x 2  
## id word   
## <chr> <chr>   
## 1 <NA> global   
## 2 <NA> landslide   
## 3 <NA> catalog   
## 4 <NA> glc   
## 5 <NA> developed   
## 6 <NA> goal   
## 7 <NA> identifying  
## 8 <NA> rainfall   
## 9 <NA> triggered   
## 10 <NA> landslide   
## # ... with 3,734,563 more rows

### 8.1.2 Some initial simple exploration

What are the most common words in the NASA dataset titles? We can use count() from dplyr to check this out.

nasa\_title %>%   
 count(word,sort=T)

## # A tibble: 17,552 x 2  
## word n  
## <chr> <int>  
## 1 phase 8291  
## 2 data 4735  
## 3 1 4110  
## 4 level 2816  
## 5 ii 2725  
## 6 global 2445  
## 7 2 1993  
## 8 daily 1776  
## 9 3 1773  
## 10 v1.0 1767  
## # ... with 17,542 more rows

What about the descriptions?

nasa\_desc %>%   
 count(word,sort=TRUE)

## # A tibble: 51,018 x 2  
## word n  
## <chr> <int>  
## 1 data 92394  
## 2 modis 27715  
## 3 global 25948  
## 4 system 22782  
## 5 2 21956  
## 6 1 20448  
## 7 product 20079  
## 8 surface 17725  
## 9 resolution 17601  
## 10 earth 17015  
## # ... with 51,008 more rows

Words like “data??? and “global??? are used very often in NASA titles and descriptions. We may want to remove digits and some “words??? like “v1??? from these data frames for many types of analyses; they are not too meaningful for most audiences.

We can do this by making a list of custom stop words and using anti\_join() to remove them from the data frame, just like we removed the default stop words that are in the tidytext package. This approach can be used in many instances and is a great tool to bear in mind.

my\_stopwords <- data\_frame(word=c(as.character(1:10),  
 "v1","v03","l2","l3","v.5.2.0",  
 "v003","v004","v005","v996","v7"))  
nasa\_title <- nasa\_title %>%   
 anti\_join(my\_stopwords)

## Joining, by = "word"

nasa\_desc <- nasa\_desc %>%   
 anti\_join(my\_stopwords)

## Joining, by = "word"

What are the most common keywords?

nasa\_keyword %>%   
 group\_by(keyword) %>%   
 count(sort=TRUE)

## # A tibble: 7,100 x 2  
## # Groups: keyword [7,100]  
## keyword n  
## <chr> <int>  
## 1 EARTH SCIENCE 18128  
## 2 Completed 8937  
## 3 ATMOSPHERE 8574  
## 4 Oceans 8074  
## 5 Ocean Optics 7524  
## 6 Ocean Color 7289  
## 7 National Geospatial Data Asset 7072  
## 8 NGDA 7072  
## 9 ATMOSPHERIC WATER VAPOR 3610  
## 10 LAND SURFACE 3495  
## # ... with 7,090 more rows

We likely want to change all of the keywords to either lower or uppoer case to get rid of duplicates like “OCEANS” and “Oceans”. Let’s do that here.

nasa\_keyword <- nasa\_keyword %>%   
 mutate(keyword=toupper(keyword))

## 8.2 Word co-occurences and correlations

As a next step, let’s examine which words commonly occur together in the titles, descriptions, and keywords of NASA datasets, as described in Chapter 4. We can the nexamine word networks of these fields; this may help us see, for example, which datasets are related to each other.

### 8.2.1 Networks of Description and title Words

We can use pairwise\_count() from windyr package to count how many times each pair of words occurs together in a title or description field.

nasa\_title

## # A tibble: 285,520 x 2  
## id word   
## <chr> <chr>   
## 1 <NA> global   
## 2 <NA> landslide   
## 3 <NA> catalog   
## 4 <NA> export   
## 5 <NA> nasa   
## 6 <NA> air   
## 7 <NA> traffic   
## 8 <NA> management  
## 9 <NA> ontology   
## 10 <NA> atmonto   
## # ... with 285,510 more rows

title\_word\_pairs <- nasa\_title %>%   
 pairwise\_count(id,word, sort = TRUE, upper = FALSE)

## Warning: Trying to compute distinct() for variables not found in the data:  
## - `row\_col`, `column\_col`  
## This is an error, but only a warning is raised for compatibility reasons.  
## The operation will return the input unchanged.

title\_word\_pairs

## # A tibble: 13,228,391 x 3  
## item1 item2 n  
## <chr> <chr> <dbl>  
## 1 55942a6ac63a7fe59b4969be 55942aaec63a7fe59b49a13f 19  
## 2 55942a6ac63a7fe59b4969be 55942aaec63a7fe59b49a140 19  
## 3 55942aaec63a7fe59b49a13f 55942aaec63a7fe59b49a140 19  
## 4 55942a6ac63a7fe59b4969be 55942aaec63a7fe59b49a141 19  
## 5 55942aaec63a7fe59b49a13f 55942aaec63a7fe59b49a141 19  
## 6 55942aaec63a7fe59b49a140 55942aaec63a7fe59b49a141 19  
## 7 <NA> 55942acbc63a7fe59b49b832 19  
## 8 <NA> 55942acbc63a7fe59b49b833 19  
## 9 55942acbc63a7fe59b49b832 55942acbc63a7fe59b49b833 19  
## 10 <NA> 55942acbc63a7fe59b49b834 19  
## # ... with 13,228,381 more rows

These are the pairs of words that occur together most often in title fields. Some of these words are obviously acronyms used within NASA, and we see how often words like “project??? and “system??? are used.

library(widyr)  
desc\_word\_pairs <- nasa\_desc %>%   
 pairwise\_count(word, id, sort = TRUE, upper = FALSE)  
nasa\_desc  
desc\_word\_pairs

These are the pairs of words that occur together most often in description fields. “Data” is a very common word in description fileds; there is no shortage of data in the datasets at NASA!

Let’s plot networks of these co-occurring words so we can see these relationships better in Figure 8.1. We will again use the ggraph package for visualizing our networks.

library(ggplot2)  
library(igraph)  
library(ggraph)  
  
set.seed(1234)  
title\_word\_pairs %>%  
 filter(n >= 250) %>%  
 graph\_from\_data\_frame() %>%  
 ggraph(layout = "fr") +  
 geom\_edge\_link(aes(edge\_alpha = n, edge\_width = n), edge\_colour = "cyan4") +  
 geom\_node\_point(size = 5) +  
 geom\_node\_text(aes(label = name), repel = TRUE,   
 point.padding = unit(0.2, "lines")) +  
 theme\_void()

We see some clear clustering in this network of title words;words in NASA dataset titles are largely organized into several families of words that tend to go together.

What about the words from the description fields?

set.seed(1234)  
desc\_word\_pairs %>%   
 filter(n>=5000) %>%   
 graph\_from\_data\_frame() %>%   
 ggraph(layout="fr")+  
 geom\_edge\_link(aes(edge\_alpha=n,edge\_width=n),edge\_colour="darkred")+  
 geom\_node\_point(size=5)+  
 geom\_node\_text(aes(label=name),repel=TRUE,  
 point.padding=unit(0.2,"lines"))+  
 theme\_void()

Figure 8.2 shows such *strong* connections between the top dozen or so words (words like “data???, “global???, “resolution???, and “instrument???) that we do not see clear clustering structure in the network. We may want to use tf-idf (as described in detail in Chapter 3) as a metric to find characteristic words for each description field, instead of looking at counts of words.

### 8.2.2 Networks of Keywords

Next, let’s make a network of the keywords in Figure 8.3 to see which keywords commonly occur together in the same datasets:

keyword\_pairs <- nasa\_keyword %>%   
 pairwise\_count(keyword, id, sort = TRUE, upper = FALSE)

## Warning: Trying to compute distinct() for variables not found in the data:  
## - `row\_col`, `column\_col`  
## This is an error, but only a warning is raised for compatibility reasons.  
## The operation will return the input unchanged.

keyword\_pairs

## # A tibble: 14,957,167 x 3  
## item1 item2 n  
## <chr> <chr> <dbl>  
## 1 OCEANS OCEAN OPTICS 7323  
## 2 OCEANS OCEAN COLOR 7271  
## 3 OCEAN OPTICS OCEAN COLOR 7271  
## 4 EARTH SCIENCE ATMOSPHERE 6535  
## 5 EARTH SCIENCE ATMOSPHERIC WATER VAPOR 2856  
## 6 ATMOSPHERE ATMOSPHERIC WATER VAPOR 2856  
## 7 EARTH SCIENCE OCEANS 2676  
## 8 EARTH SCIENCE LAND SURFACE 2174  
## 9 EARTH SCIENCE ATMOSPHERIC TEMPERATURE 1972  
## 10 ATMOSPHERE ATMOSPHERIC TEMPERATURE 1972  
## # ... with 14,957,157 more rows

library(igraph)

##   
## Attaching package: 'igraph'

## The following objects are masked from 'package:lubridate':  
##   
## %--%, union

## The following objects are masked from 'package:dplyr':  
##   
## as\_data\_frame, groups, union

## The following objects are masked from 'package:purrr':  
##   
## compose, simplify

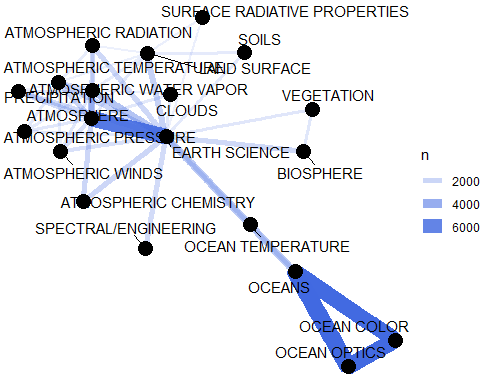
## The following object is masked from 'package:tidyr':  
##   
## crossing

## The following object is masked from 'package:tibble':  
##   
## as\_data\_frame

## The following objects are masked from 'package:stats':  
##   
## decompose, spectrum

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

set.seed(1234)  
keyword\_pairs %>%  
 filter(n >= 700) %>%  
 graph\_from\_data\_frame() %>%  
 ggraph(layout = "fr") +  
 geom\_edge\_link(aes(edge\_alpha = n, edge\_width = n), edge\_colour = "royalblue") +  
 geom\_node\_point(size = 5) +  
 geom\_node\_text(aes(label = name), repel = TRUE,  
 point.padding = unit(0.2, "lines")) +  
 theme\_void()



We definitely see clustering here, and strong connections between keywords like “OCEANS???, “OCEAN OPTICS???, and “OCEAN COLOR???, or “PROJECT??? and “COMPLETED???.

To examine the relationships among keywords in a different way, we can find the correlation among the keywords as described in Chapter 4. This looks for those keywords that are more likely to occur together than with other keywords in a description field.

library(tidyverse)  
library(magrittr)  
library(tidytext)  
  
keyword\_cors <- nasa\_keyword %>%   
 group\_by(keyword) %>%   
 filter(n() >=50) %>%   
 pairwise\_cor(keyword,id,sort=TRUE,upper=FALSE)

## Warning: Trying to compute distinct() for variables not found in the data:  
## - `row\_col`, `column\_col`  
## This is an error, but only a warning is raised for compatibility reasons.  
## The operation will return the input unchanged.

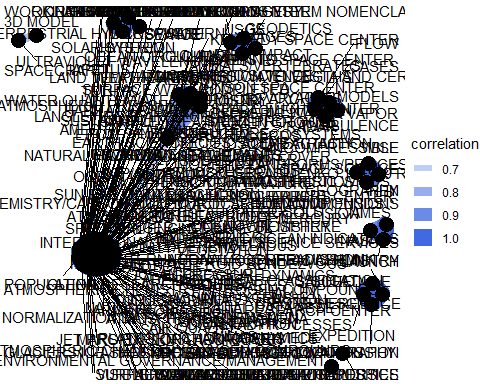
keyword\_cors

## # A tibble: 20,503 x 3  
## item1 item2 correlation  
## <chr> <chr> <dbl>  
## 1 KNOWLEDGE SHARING 1.  
## 2 DASHLINK AMES 1.  
## 3 NUCLEIC ACID EXTRACTION NUCLEIC ACID SEQUENCING 1.  
## 4 WEATHER EVENTS SAMPLE COLLECTION 1.  
## 5 NUCLEIC ACID EXTRACTION SAMPLE COLLECTION 1.  
## 6 NGDA GRAVITY/GRAVITATIONAL FIELD 1.  
## 7 NATIONAL GEOSPATIAL DATA ASSET GRAVITY/GRAVITATIONAL FIELD 1.  
## 8 NUCLEIC ACID EXTRACTION GRAVITY/GRAVITATIONAL FIELD 1.  
## 9 LIBRARY CONSTRUCTION SUSTAINABILITY 1.  
## 10 LIBRARY CONSTRUCTION SOLAR WIND 1.  
## # ... with 20,493 more rows

Notice that these keywords at the top of this sorted data frame have correlation coefficients equal to 1; they always occur together. This means these are redundant keywords. It may not make sense to continue to use both of the keywords in these sets of pairs; instead, just one keyword could be used.

Let’s visualize the network of keyword correlations, just as we did for keyword co-occurences.

set.seed(1234)  
keyword\_cors %>%   
 filter(correlation>.6) %>%   
 graph\_from\_data\_frame() %>%   
 ggraph(layout="fr")+  
 geom\_edge\_link(aes(edge\_alpha=correlation,edge\_width=correlation),edge\_colour="royalblue")+  
 geom\_node\_point(size=5)+  
 geom\_node\_text(aes(label=name),repel=T,  
 point.padding=unit(0.2,"lines"))+  
 theme\_void()



This network in Figure 8.4 appears much different than the co-occurence network. The difference is that the co-occurrence network asks a question about which keyword pairs occur most often, and the correlation network asks a question about which keywordsoccur more often together than with other keywords. Notice here the high number of small clusters of keywords; the network structure can be extracted (for further analysis) from the graph\_from\_data\_frame() function above.

## 8.3 Calculating tf-idf for the description fields

The network graph in Figure 8.2 showed us that the description fields are dominated by a few common words like “data???, “global???, and “resolution???; this would be an excellent opportunity to use tf-idf as a statistic to find characteristic words for individual description fields. As discussed in Chapter 3, we can use tf-idf, the term frequency times inverse document frequency, to identify words that are especially important to a document within a collection of documents. Let’s apply that approach to the description fields of these NASA datasets.

### 8.3.1 What is tf-idf for the description field words?

We will consider each description field a document, and the whole set of description fields the collection or corpus of documents. We have already used unnest\_tokens() earlier in this chapter to make a tidy data frame of the words in the description fields, so now we can use bind\_tf\_idf() to calculate tf-idf for each word.

desc\_tf\_idf <- nasa\_desc %>%   
 count(id,word,sort=T) %>%   
 ungroup() %>%   
 bind\_tf\_idf(word,id,n)

What are the highest tf-idf words in the NASA description fields?

desc\_tf\_idf %>%   
 arrange(-tf\_idf)

## # A tibble: 1,378,189 x 6  
## id word n tf idf tf\_idf  
## <chr> <chr> <int> <dbl> <dbl> <dbl>  
## 1 55942a7cc63a7fe5~ rdr 1 1 9.32 9.32  
## 2 55942ac9c63a7fe5~ palsar\_radiometric\_terrain\_~ 1 1 9.32 9.32  
## 3 55942ac9c63a7fe5~ palsar\_radiometric\_terrain\_~ 1 1 9.32 9.32  
## 4 55942a7dc63a7fe5~ mri 1 1 8.92 8.92  
## 5 55942a7bc63a7fe5~ lgrs 1 1 8.22 8.22  
## 6 55942a7bc63a7fe5~ lgrs 1 1 8.22 8.22  
## 7 55942a7bc63a7fe5~ lgrs 1 1 8.22 8.22  
## 8 55942ad8c63a7fe5~ template\_proddescription 1 1 7.94 7.94  
## 9 55942ad8c63a7fe5~ template\_proddescription 1 1 7.94 7.94  
## 10 55942ad8c63a7fe5~ template\_proddescription 1 1 7.94 7.94  
## # ... with 1,378,179 more rows

These are the most important words in the description fields as measured by tf-idf, meaning they are common but not too common.

Notice we have run into an issue here; both n and term frequency are equal to 1 for these terms, meaning that these were description fields that only had a single word in them. If a description field only contains one word, the tf-idf algorithm will think that is a very important word.

Depending on our analytics goals, it might be a good idea to throw out all description fileds that have very few words.

### 8.3.2 Connecting description fields to keywords

We now know which words in the descriptions have high tf-idf, and we also have labels for these descriptions in the keywords. Let’s do a full join of the keyword data frame and the data frame of description words with tf-idf, and then find the highest tf-idf words for a given keyword.

desc\_tf\_idf <- full\_join(desc\_tf\_idf, nasa\_keyword, by = "id")

Let’s plot some of the most important words, as measured by tf-idf a few example keywords used on NASA datasets. First, let’s use dplyr operations to filter for the keywords we want to examine and take just the top 15 words for each keyword. Then, let’s plot those words in Figure 8.5.

desc\_tf\_idf %>%   
 filter(!near(tf, 1)) %>%  
 filter(keyword %in% c("SOLAR ACTIVITY", "CLOUDS",   
 "SEISMOLOGY", "ASTROPHYSICS",  
 "HUMAN HEALTH", "BUDGET")) %>%  
 arrange(desc(tf\_idf)) %>%  
 group\_by(keyword) %>%  
 distinct(word, keyword, .keep\_all = TRUE) %>%  
 top\_n(15, tf\_idf) %>%   
 ungroup() %>%  
 mutate(word = factor(word, levels = rev(unique(word)))) %>%  
 ggplot(aes(word, tf\_idf, fill = keyword)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~keyword, ncol = 3, scales = "free") +  
 coord\_flip() +  
 labs(title = "Highest tf-idf words in NASA metadata description fields",  
 caption = "NASA metadata from https://data.nasa.gov/data.json",  
 x = NULL, y = "tf-idf")

Using tf-idf has allowed us to identify important description words for each of these keywords. Datasets labeled with the keyword “SEISMOLOGY??? have words like “earthquake???, “risk???, and “hazard??? in their description, while those labeled with “HUMAN HEALTH??? have descriptions characterized by words like “wellbeing???, “vulnerability???, and “children.??? Most of the combinations of letters that are not English words are certainly acronyms (like OMB for the Office of Management and Budget), and the examples of years and numbers are important for these topics. The tf-idf statistic has identified the kinds of words it is intended to, important words for individual documents within a collection of documents.

## 8.4 Topic modeling

Using tf-idf as a statistic has already given us insight into the content of NASA description fields, but let’s try an additional approach to the question of what the NASA descriptions fields are about. We can use topic modeling as described in Chapter 6 to model each document (description field) as a mixture of topics and each topic as a mixture of words. As in earlier chapters, we will use [latent Dirichlet allocation (LDA)](https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation) for our topic modeling; there are other possible approaches for topic modeling.

### 8.4.1 Casting to a document-term matrix

To do the topic modeling as implemented here, we need to make a DocumentTermMatrix, a special kind of matrix from the tm package (of course, this is just a specific implementation of the general concept of a “document-term matrix???). Rows correspond to documents (description texts in our case) and columns correspond to terms (i.e., words); it is a sparse matrix and the values are word counts.

Let’s clean up the text a bit using stop words to remove some of the nonsense “words??? leftover from HTML or other character encoding. We can use bind\_rows() to add our custom stop words to the list of default stop words from the tidytext package, and then all at once use anti\_join() to remove them all from our data frame.

library(tidytext)  
library(tidyverse)  
  
my\_stop\_words <- bind\_rows(stop\_words,   
 data\_frame(word = c("nbsp", "amp", "gt", "lt",  
 "timesnewromanpsmt", "font",  
 "td", "li", "br", "tr", "quot",  
 "st", "img", "src", "strong",  
 "http", "file", "files",  
 as.character(1:12)),   
 lexicon = rep("custom", 30)))  
  
  
nasa\_desc  
word\_counts <- nasa\_desc %>%   
 anti\_join(my\_stop\_words) %>%   
 count(id,word,sort=TRUE) %>%   
 ungroup()  
word\_counts

This is the information we need, the number of times each word is used in each document, to make a DocumentTermMatrix. We can cast() from our tidy text format to this non-tidy format as described in detail in Chapter 5.

desc\_dtm <- word\_counts %>%   
 cast\_dtm(id,word,n)  
desc\_dtm

We see that this dataset contains documents (each of them a NASA description field) and terms (words). Notice that this example document-term matrix is (very close to) 100% sparse, meaning that almost all of the entries in this matrix are zero. Each non-zero entry corresponds to a certain word appearing in a certain document.

### 8.4.2 Ready for topic modeling

Now let’s use the [topic model](https://cran.r-project.org/web/packages/topicmodels/index.html) package to creat an LDA model. How many topics will we tell the algorithm to make? This is a question like in k-means clustering: we don’t really know ahead of time.

We tried the following modeling procedures using 8,16,24,32 and 64 topics; we found that at 24 topics, documents are still getting sorted into topics cleanly but going much beyond that caused the distributions of , the probability that each document belongs in each topic, to look worrisome. We will show more details on this later.

library(topicmodels)  
  
# data overview  
class(desc\_dtm)  
desc\_dtm  
  
# desc\_lda <- LDA(desc\_dtm,k=24,control=list(seed=1234))  
# desc\_lda

This is a stochastic algorithm that could have different results depending on where the algorithm starts, so we need to specify a seed for reproducibility as shown here.

### 8.4.3 Interpreting the topic model

Now that we have built the model, let’s tidy() the results of the model, i.e., construct a tidy data frame that summarizes the results of the model. The tidytext package includes a tidying method for LDA models from the topicmodels package.

tidy\_lda <- tidy(desc\_lda)  
tidy\_lda

The column tells us the probability of that term being generated from that topic for that document. It is the probability of that term (word) belonging to that topic. Notice that some of the values for are very, very low, and some are not so low.

What is each topic about? Let’s examine the top 10 terms for each topic.

top\_terms <- tidy\_lda %>%   
 group\_by(topic) %>%   
 top\_n(10,beta) %>%   
 ungroup() %>%   
 arrange(topic,-beta)  
top\_terms

It is not very easy to interpret what the topics are about from a data frame like this so let’s look at this information visually in Figure 8.6.

top\_terms %>%  
 mutate(term = reorder(term, beta)) %>%  
 group\_by(topic, term) %>%   
 arrange(desc(beta)) %>%   
 ungroup() %>%  
 mutate(term = factor(paste(term, topic, sep = "\_\_"),   
 levels = rev(paste(term, topic, sep = "\_\_")))) %>%  
 ggplot(aes(term, beta, fill = as.factor(topic))) +  
 geom\_col(show.legend = FALSE) +  
 coord\_flip() +  
 scale\_x\_discrete(labels = function(x) gsub("\_\_.+$", "", x)) +  
 labs(title = "Top 10 terms in each LDA topic",  
 x = NULL, y = expression(beta)) +  
 facet\_wrap(~ topic, ncol = 4, scales = "free")

We can see what a dominant word “data” is in these description texts. In addition, there are meaningful differences between these collections of terms, from terms about soil, forests, and biomass in topic 12 to terms about design, systems, and technology in topic 21. The topic modeling process has identified groupings of terms that we can understand as human readers of these description fields.

We just explored which words are associated with which topics. Next, let’s examine which topics are associated with which description fields (i.e., documents). We will look at a different probability for this, ., the probability that each document belongs in each topic, again using the tidy verb.

lda\_gamma <- tidy(desc\_lda, matrix = "gamma")  
lda\_gamma

Notice that some of the probabilites visible at the top of the data frame are low and some are higher. Our model has assigned a probability to each description belonging to each of the topics we constructed from the sets of words. How are the probabilities distributed? Let’s visualize them (Figure 8.7).

ggplot(lda\_gamma, aes(gamma)) +  
 geom\_histogram() +  
 scale\_y\_log10() +  
 labs(title = "Distribution of probabilities for all topics",  
 y = "Number of documents", x = expression(gamma))

First notice that the y-axis is plotted on a log scale; otherwise it is difficult to make out any detail in the plot. Next, notice that  
 runs from 0 to 1; remember that this is the probability that a given document belongs in a given topic. There are many values near zero, which means there are many documents that do not belong in each topic. Also, there are many values near ; these are the documents that do belong in those topics. This distribution shows that documents are being well discriminated as belonging to a topic or not. We can also look at how the probabilities are distributed within each topic, as shown in Figure 8.8.

ggplot(lda\_gamma, aes(gamma, fill = as.factor(topic))) +  
 geom\_histogram(show.legend = FALSE) +  
 facet\_wrap(~ topic, ncol = 4) +  
 scale\_y\_log10() +  
 labs(title = "Distribution of probability for each topic",  
 y = "Number of documents", x = expression(gamma))

### 8.4.4 Connecting topic modeling with keywords

let’s connect these topic models with the keywords and see what relatinships we can find. We can full\_join() this to the human-tagged keywords and discover which keywords are associated with which topic.

lda\_gamma <- full\_join(lda\_gamma, nasa\_keyword, by = c("document" = "id"))  
  
lda\_gamma

Now we can use filter() to keep only the document-topic entries that have probabilities greater than some cut-off value; let’s use 0.9.

top\_keywords <- lda\_gamma %>%   
 filter(gamma>0.9) %>%   
 count(topic,keyword,sort=TRUE)  
  
top\_keywords

What are the top keywords for each topic?

top\_keywords %>%  
 group\_by(topic) %>%  
 top\_n(5, n) %>%  
 group\_by(topic, keyword) %>%  
 arrange(desc(n)) %>%   
 ungroup() %>%  
 mutate(keyword = factor(paste(keyword, topic, sep = "\_\_"),   
 levels = rev(paste(keyword, topic, sep = "\_\_")))) %>%  
 ggplot(aes(keyword, n, fill = as.factor(topic))) +  
 geom\_col(show.legend = FALSE) +  
 labs(title = "Top keywords for each LDA topic",  
 x = NULL, y = "Number of documents") +  
 coord\_flip() +  
 scale\_x\_discrete(labels = function(x) gsub("\_\_.+$", "", x)) +  
 facet\_wrap(~ topic, ncol = 4, scales = "free")

et’s take a step back and remind ourselves what Figure 8.9 is telling us. NASA datasets are tagged with keywords by human beings, and we have built an LDA topic model (with 24 topics) for the description fields of the NASA datasets. This plot answers the question, “For the datasets with description fields that have a high probability of belonging to a given topic, what are the most common human-assigned keywords?”

It’s interesting that the keywords for topics 13, 16, and 18 are essentially duplicates of each other (“OCEAN COLOR”, “OCEAN OPTICS”, “OCEANS”), because the top words in those topics do exhibit meaningful differences, as shown in Figure 8.6. Also note that by number of documents, the combination of 13, 16, and 18 is quite a large percentage of the total number of datasets represented in this plot, and even more if we were to include topic 11. By number, there are many datasets at NASA that deal with oceans, ocean color, and ocean optics. We see “PROJECT COMPLETED” in topics 9, 10, and 21, along with the names of NASA laboratories and research centers. Other important subject areas that stand out are groups of keywords about atmospheric science, budget/finance, and population/human dimensions. We can go back to Figure 8.6 on terms and topics to see which words in the description fields are driving datasets being assigned to these topics. For example, topic 4 is associated with keywords about population and human dimensions, and some of the top terms for that topic are “population”, “international”, “center”, and “university”.

## 8.5 Summary

By using a combination of network analysis, tf-idf, and topic modeling, we have come to a greater understanding of how datasets are related at NASA. Specifically, we have more information now about how keywords are connected to each other and which datasets are likely to be related. The topic model could be used to suggest keywords based on the words in the description field, or the work on the keywords could suggest the most important combination of keywords for certain areas of study.