DSI Summer Workshops Series

July 26, 2018

Peggy Lindner
Center for Advanced Computing & Data Science (CACDS)
Data Science Institute (DSI)
University of Houston
plindner@uh.edu

Please make sure you have Jupyterhub running with support for R and all the required packages installed. Data for this and other tutorials can be found in the github repsoitory for the Summer 2018 DSI Workshops https://github.com/peggylind/Materials_Summer2018 (https://github.com/peggylind/Materials_Summer2018)

Data Mining Twitter Data

Understand basics of twitter data mining using R

Twitter

- An online social networking service that enables users to send and read short 140-character messages called "tweets" (Wikipedia)
- Over 300 million monthly active users (as of 2015)
- Creating over 500 million tweets per day

Techniques

- Text Mining
- Topic Modeling
- Sentiment Analysis

Tools

- Twitter API
- R and specifically the following packages
- [twitteR] Twitter data extraction
- tm (https://cran.r-project.org/web/packages/tm/vignettes/tm.pdf) Text cleaning mining
- topicmodels (https://www.tidytextmining.com/topicmodeling.html) Topic Modeling

...

Visualization

- ggplot2 (http://ggplot2.tidyverse.org/) Modern R visulaizations
- wordcloud (http://developer.marvel.com) Make some nice word clouds
- RColorBrewer (https://dataset.readthedocs.org/en/latest/) Get color into your visualizations

...

Process

1. Extract tweets and followers from the Twitter website with R and the twitteR package
2. With the tm package, clean text by removing punctuations, numbers, hyperlinks and stop words, followed by stemming and stem completion
3. Build a term-document matrix
4. Analyse topics with the topicmodels package
5. Analyse sentiment with the sentiment140 package

6. Analyse following/followed and retweeting relationships with the igraph package

Using existing twitter data within this tutorial

`# you could download Twitter data manually from site

url <- "http://www.rdatamining.com/data/RDataMining-Tweets-20160212.rds (http://www.rdatamining.com/data/RDataMining-Tweets-20160212.rds)"

download.file(url, destfile = "RDataMining-Tweets-20160212.rds") `

Retrieve Tweets

In [35]:

```
#load the twitteR library
library(twitteR)
```

A) using the Twitter API

The following code is merely an abstract example. You will have to learn more about the Twitter API and how to use it at: Twitter Developer (https://developer.twitter.com/en.html)

And prepare you Twitter account: https://towardsdatascience.com/setting-up-twitter-for-text-mining-in-r-bcfc5ba910f4 (https://towardsdatascience.com/setting-up-twitter-for-text-mining-in-r-bcfc5ba910f4)

In [36]:

```
# This code will not run!
# Change the next four lines based on your own consumer_key, c
onsume_secret, access_token, and access_secret.
consumer_key <- "dfgbfdbhe"
consumer_secret <- "fdbdbh"
access_token <- "dfbhdf"
access_secret <- "fbhfd"

setup_twitter_oauth(consumer_key, consumer_secret, access_toke
n, access_secret)
tw = twitteR::searchTwitter('#realDonaldTrump + #HillaryClinto
n', n = 1e4, since = '2016-11-08', retryOnRateLimit = 1e3)
d = twitteR::twListToDF(tw)
```

```
[1] "Using direct authentication"
```

```
Error in check_twitter_oauth(): OAuth authenticatio
n error:
This most likely means that you have incorrectly ca
lled setup_twitter_oauth()'
Traceback:

1. setup_twitter_oauth(consumer_key, consumer_secre
t, access_token,
 .     access_secret)
2. check_twitter_oauth()
3. stop("OAuth authentication error:\nThis most lik
ely means that you have incorrectly called setup_tw
itter_oauth()'")
```

B) load from file

In [ ]:

```
#read the data from a file
tweets <- readRDS("dataJuly26th/RDataMining-Tweets-20160212.rd
s")

#let's look what we got
tweets
```

Let's explore

In [38]:

```
# number of tweets in dataset
(n.tweet <- length(tweets))
```

448

In [39]:

```
# convert to data frame
tweets.df <- twListToDF(tweets)
```

In [40]:

```
# look at tweet #190
tweets.df[190, c("id", "created", "screenName", "replyToSN",
"favoriteCount", "retweetCount", "longitude", "latitude", "tex
t")]
```

	id	created	screenName	replyToSN	fa
190	362866933894352898	2013- 08-01 09:26:33	RDataMining	NA	9

Text Cleaning

In [41]:

```
# we will use the tm library
library(tm)
```

In [42]:

```
# build a corpus, and specify the source to be character vecto
rs
myCorpus <- Corpus(VectorSource(tweets.df$text))

#what did we just create?
myCorpus
```

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (inde
xed): 0
Content:  documents: 448
```

In [43]:

```
# print tweet # and make text fit for slide width
writeLines(strwrap(tweets.df$text[3], 60))
```

```
Thanks to all for your ongoing support to
https://t.co/TrMJxOBX73. Merry Christmas and Happy
New
Year!
```

In [44]:

```r
# print tweet #190 and make text fit for slide width
writeLines(strwrap(tweets.df$text[190], 60))

# convert to lower case
myCorpus <- tm_map(myCorpus, content_transformer(tolower))

writeLines(strwrap(myCorpus[[190]]$content, 60))
```

```
The R Reference Card for Data Mining now provides l
inks to
packages on CRAN. Packages for MapReduce and Hadoop
added.
http://t.co/RrFypol8kw
the r reference card for data mining now provides l
inks to
packages on cran. packages for mapreduce and hadoop
added.
http://t.co/rrfypol8kw
```

In [45]:

```r
# remove URLs
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeURL))

writeLines(strwrap(myCorpus[[190]]$content, 60))
```

```
the r reference card for data mining now provides l
inks to
packages on cran. packages for mapreduce and hadoop
added.
```

In [46]:

```
# remove anything other than English letters or space
removePunct <- function(x) gsub("[^[:alpha:][:space:]]*", "",
 x)
myCorpus <- tm_map(myCorpus, content_transformer(removePunct))
myCorpus <- tm_map(myCorpus, removeNumbers)

writeLines(strwrap(myCorpus[[190]]$content, 60))
```

```
the r reference card for data mining now provides l
inks to
packages on cran packages for mapreduce and hadoop
added
```

In [47]:

```
# remove stopwords
myStopwords <- c(setdiff(stopwords('english'), c("r", "big")),
                 "use", "see", "used", "via", "amp")
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)

writeLines(strwrap(myCorpus[[190]]$content, 60))
```

```
r reference card data mining now provides links pac
kages
cran packages mapreduce hadoop added
```

In [48]:

```
# remove extra whitespace
myCorpus <- tm_map(myCorpus, stripWhitespace)

writeLines(strwrap(myCorpus[[190]]$content, 60))
```

```
r reference card data mining now provides links pac
kages
cran packages mapreduce hadoop added
```

In [54]:

```
# keep a copy for stem completion later
myCorpusCopy <- myCorpus
```

In [55]:

```
myCorpusCopy <- tm_map(myCorpusCopy, stemDocument) # stem word
s
writeLines(strwrap(myCorpusCopy[[190]]$content, 60))
```

```
r refer card data mine now provid link packag cran
packag
mapreduc hadoop ad
```

In [56]:

```
stemCompletion2 <- function(x, dictionary) {
  x <- unlist(strsplit(as.character(x), " "))
  x <- x[x != ""]
  x <- stemCompletion(x, dictionary=dictionary)
  x <- paste(x, sep="", collapse=" ")
  PlainTextDocument(stripWhitespace(x))
}
myCorpusCopy <- lapply(myCorpusCopy, stemCompletion2, dictiona
ry=myCorpusCopy)
myCorpusCopy <- Corpus(VectorSource(myCorpusCopy))
writeLines(strwrap(myCorpusCopy[[190]]$content, 60))
```

```
list(content = "r refer card data mine now provid l
ink
packag cran packag mapreduc hadoop ad", meta = list
(author
= character(0), datetimestamp = list(sec =
32.4582738876343, min = 10, hour = 3, mday = 26, mo
n = 6,
year = 118, wday = 4, yday = 206, isdst = 0), descr
iption =
character(0), heading = character(0), id = characte
r(0),
language = character(0), origin = character(0)))
```

Issues in Stem completion

In [57]:

```
# let's count some words to see what is going on with this ste
mming
wordFreq <- function(corpus, word) {
  results <- lapply(corpus,
                    function(x) { grep(as.character(x), patter
n=paste0("\\<",word)) }
  )
  sum(unlist(results))
}
n.miner <- wordFreq(myCorpusCopy, "miner")
n.mining <- wordFreq(myCorpusCopy, "mining")
cat(n.miner, n.mining)
```

```
9 2
```

In [58]:

```
# solution: replace oldword with newword (to fix stemming issu
e)
replaceWord <- function(corpus, oldword, newword) {
  tm_map(corpus, content_transformer(gsub),
         pattern=oldword, replacement=newword)
}
myCorpus <- replaceWord(myCorpus, "miner", "mining")
myCorpus <- replaceWord(myCorpus, "universidad", "university")
myCorpus <- replaceWord(myCorpus, "scienc", "science")

writeLines(strwrap(myCorpus[[190]]$content, 60))
```

```
r reference card data mining now provides links pac
kages
cran packages mapreduce hadoop added
```

Finally! Ready to Build a document term matrix

In [49]:

```
tdm <- TermDocumentMatrix(myCorpus,
                          control = list(wordLengths = c(1, Inf)))
tdm

# look at document term matrix
idx <- which(dimnames(tdm)$Terms %in% c("r", "data", "mining"))
as.matrix(tdm[idx, 21:30])
```

```
<<TermDocumentMatrix (terms: 1298, documents: 448)>>
Non-/sparse entries: 3677/577827
Sparsity           : 99%
Maximal term length: 23
Weighting          : term frequency (tf)
```

	21	22	23	24	25	26	27	28	29	30
mining	0	0	0	0	1	0	0	0	0	1
data	0	1	0	0	1	0	0	0	0	1
r	1	1	1	1	0	1	0	1	1	1

Let's look at the Top Frequent Terms

In [50]:

```
(freq.terms <- findFreqTerms(tdm, lowfreq = 20))

# sum up the document term matrix by rows
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq >= 20)
# prepare sums for plotting
df <- data.frame(term = names(term.freq), freq = term.freq)

df
```

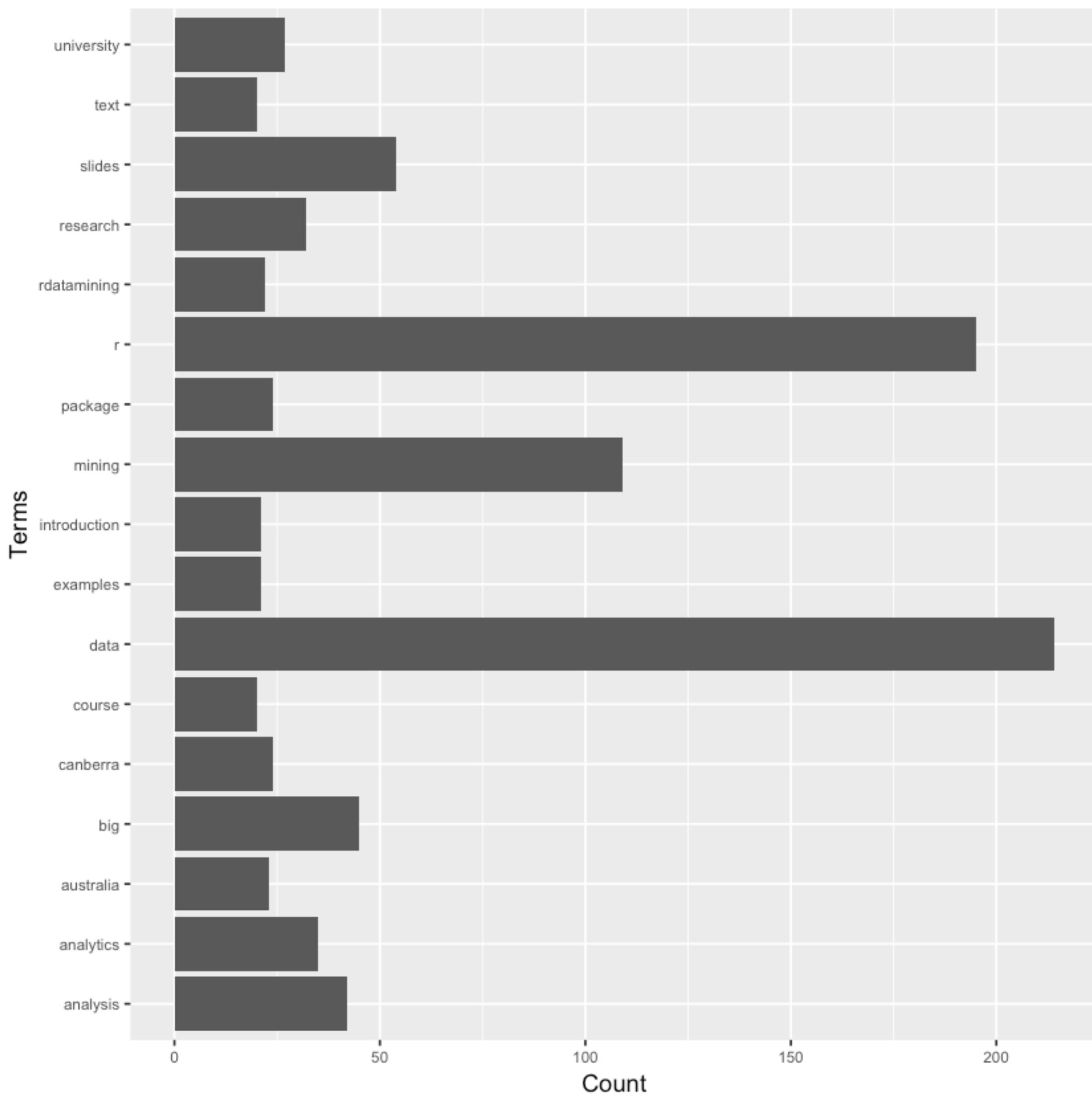
'mining' 'rdatamining' 'text' 'analytics' 'australia' 'data' 'canberra' 'university' 'slides' 'big' 'r' 'course' 'introduction' 'package' 'analysis' 'research' 'examples'

	term	freq
mining	mining	109
rdatamining	rdatamining	22
text	text	20
analytics	analytics	35
australia	australia	23
data	data	214
canberra	canberra	24
university	university	27
slides	slides	54
big	big	45
r	r	195
course	course	20
introduction	introduction	21
package	package	24
analysis	analysis	42
research	research	32
examples	examples	21

And visualize those results

In [51]:

```
#create histogram of word frequencies
library(ggplot2)
ggplot(df, aes(x=term, y=freq)) + geom_bar(stat="identity") +
  xlab("Terms") + ylab("Count") + coord_flip() +
  theme(axis.text=element_text(size=7))
```



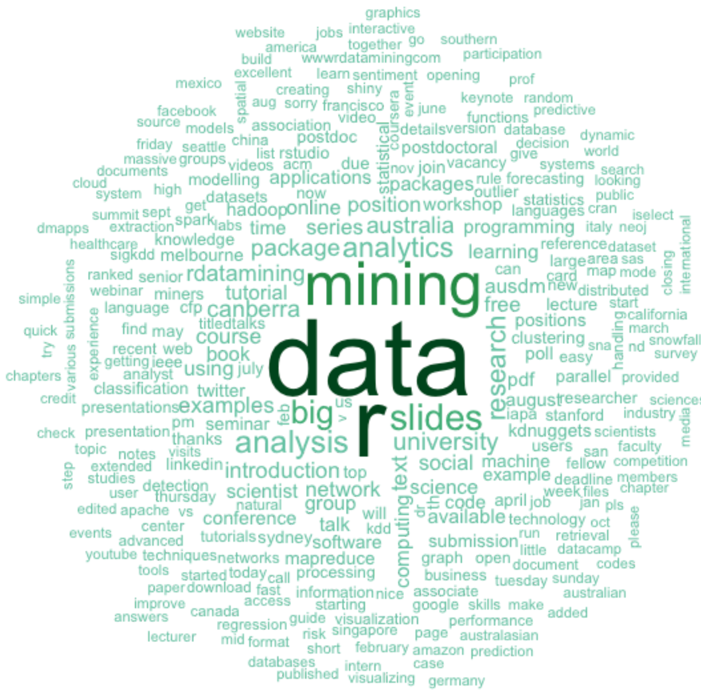
Want something more colorful and playful?

In [ ]:

```
#prep
m <- as.matrix(tdm)
# calculate the frequency of words and sort it by frequency
word.freq <- sort(rowSums(m), decreasing = T)

word.freq
# colors
library(RColorBrewer)
pal <- brewer.pal(9, "BuGn")[-(1:4)]

pal
```

In [53]:

```
# plot word cloud
library(wordcloud)
wordcloud(words = names(word.freq), freq = word.freq, min.freq
 = 3,
          random.order = F, colors = pal)
```


data
r
mining
slides
analysis
big
analytics
research
university
package
australia
canberra
rdatamining
examples
introduction
packages
position
online
applications
course
tutorial
learning
programming
social
network
science
computing
text
conference
scientist
group
talk
free
ausdm
series
workshop
book
using
time
hadoop
postdoctoral
available
submission
software
mapreduce
clustering
positions
kdnuggets
seminar
melbourne
sydney
machine
example
code
twitter
classification
statistical

Word Associations

Another way to think about word relationships is with the findAssocs() function in the tm package. For any given word, findAssocs() calculates its correlation with every other word in a TDM or DTM. Scores range from 0 to 1. A score of 1 means that two words always appear together in documents, while a score approaching 0 means the terms seldom appear in the same document.

Keep in mind the calculation for findAssocs() is done at the document level. So for every document that contains the word in question, the other terms in those specific documents are associated. Documents without the search term are ignored.

In [59]:

```
# which words are associated with 'r'?
findAssocs(tdm, "r", 0.2)

findAssocs(tdm, "data", 0.2)
```

$r =
code
0.23
users
0.21
series
0.21
markdown
0.2

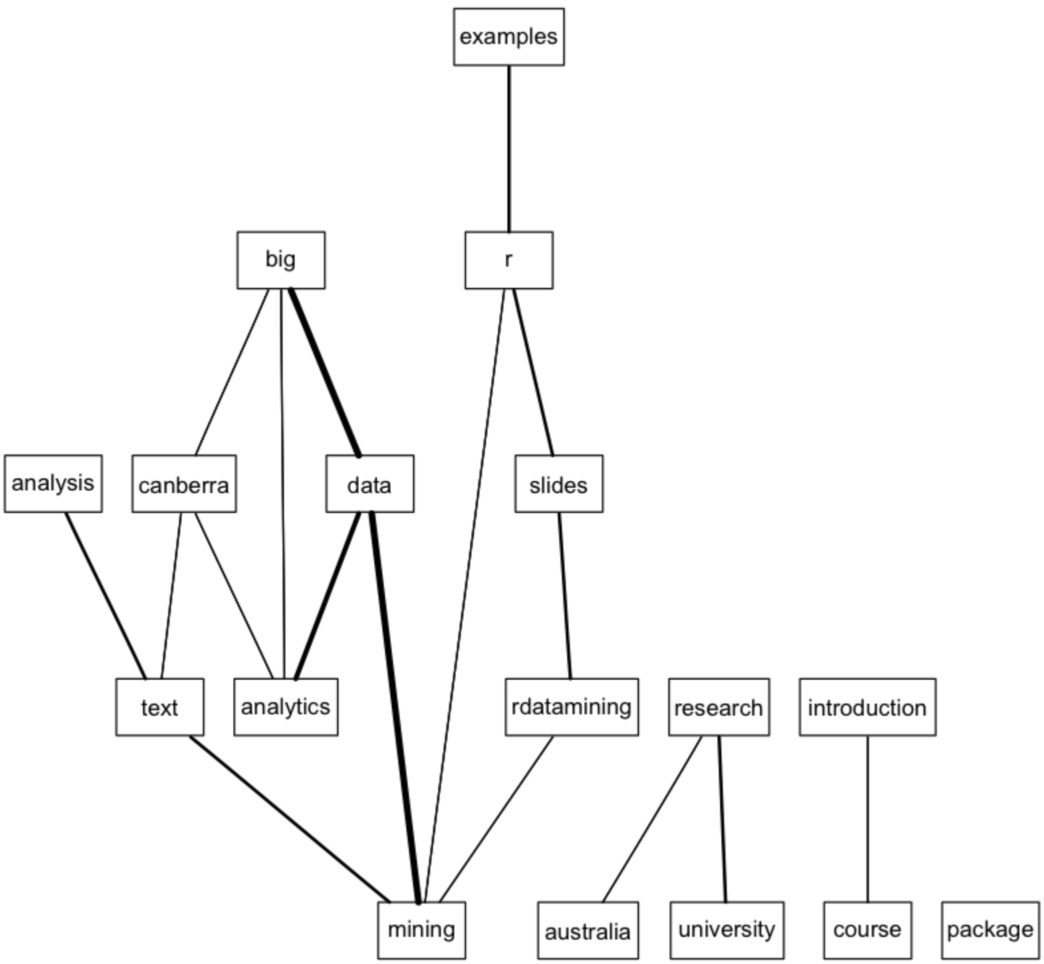
$data =
mining
0.44
big
0.44
analytics
0.31
science
0.29
poll
0.24

Network of terms

Once a few interesting term correlations have been identified, it can be useful to visually represent term correlations using the plot() function. By default the plot() function will default to a handful of randomly chosen terms, with the chosen correlation threshold, e.g.:

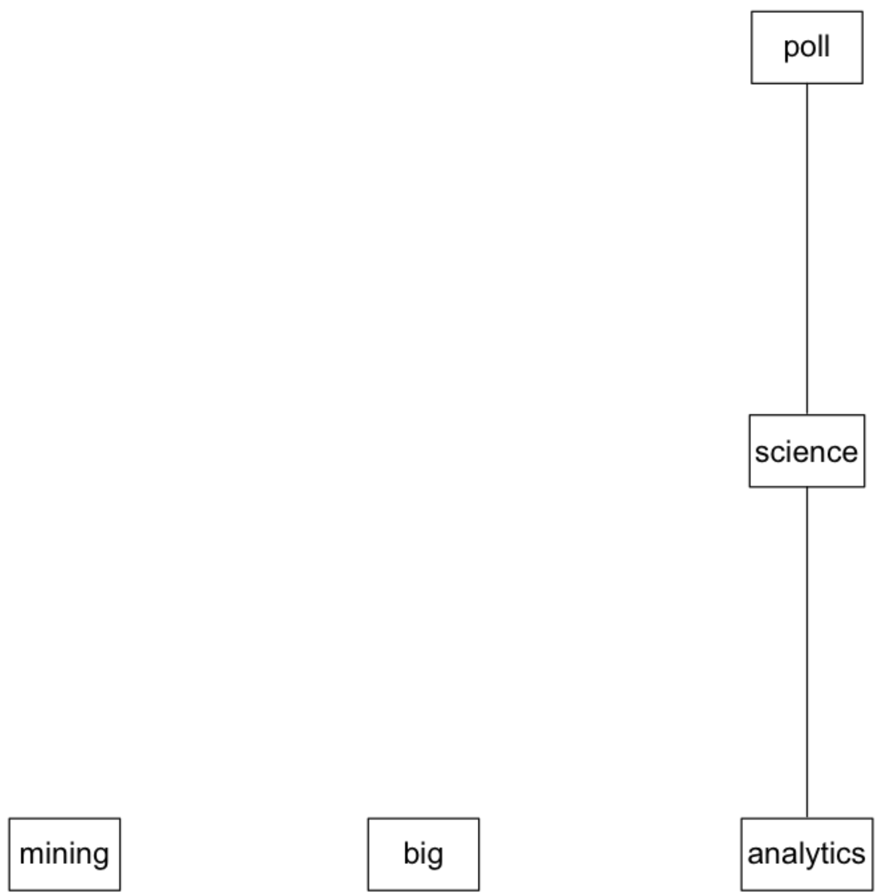
In [69]:

```
# network of terms
library(graph)
library(Rgraphviz)
plot(tdm, term = freq.terms, corThreshold = 0.1, weighting = T
)
```



In [67]:

```
plot(tdm, terms = names(findAssocs(tdm,term="data",0.2)[["dat
a"]]), corThreshold = 0.3)
```



Topic Modeling

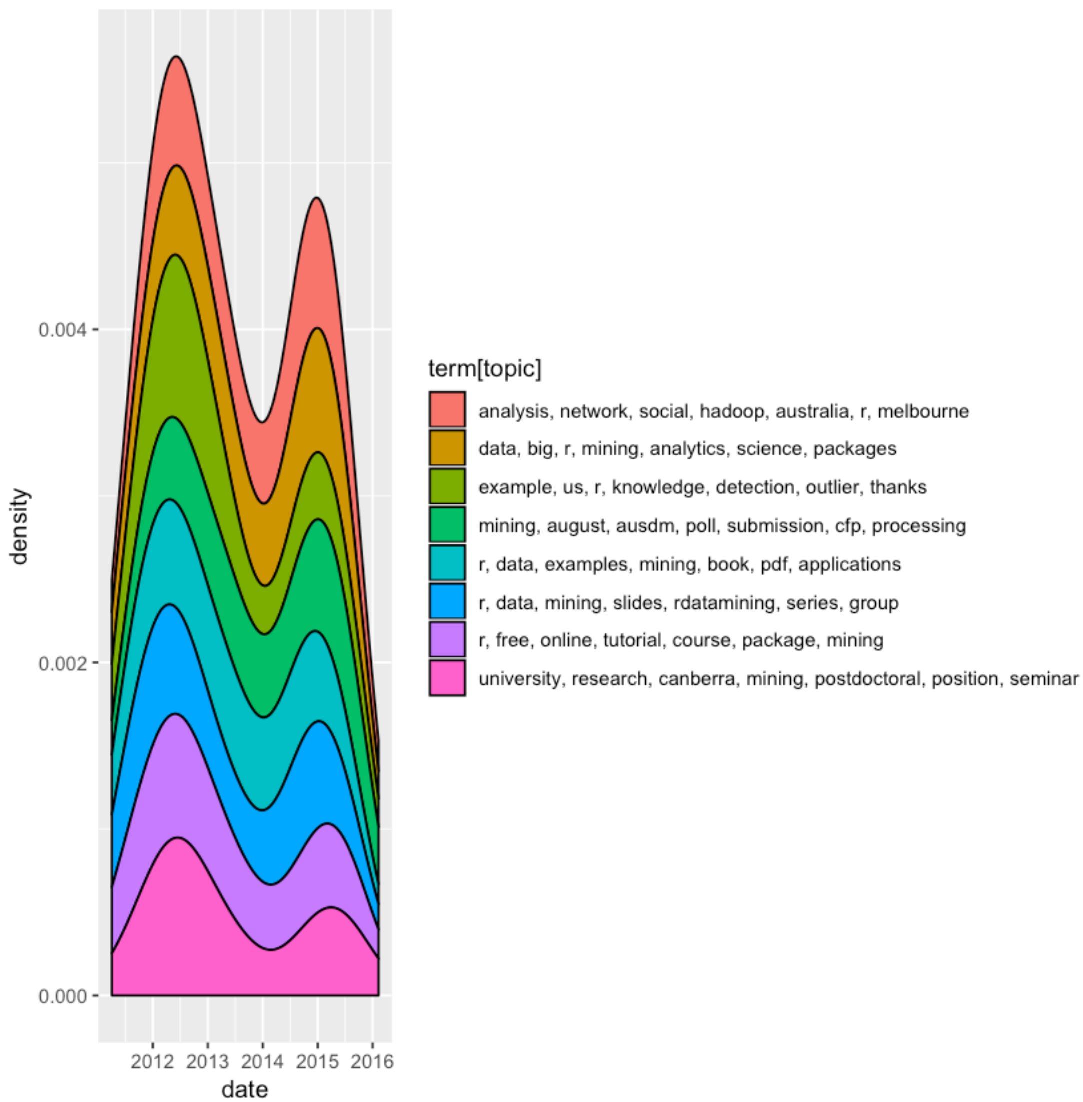
In [70]:

```
dtm <- as.DocumentTermMatrix(tdm)
library(topicmodels)
lda <- LDA(dtm, k = 8) # find 8 topics
term <- terms(lda, 7) # first 7 terms of every topic
(term <- apply(term, MARGIN = 2, paste, collapse = ", "))
```

Topic 1
'r, data, examples, mining, book, pdf, applications'
Topic 2
'data, big, r, mining, analytics, science, packages'
Topic 3
'analysis, network, social, hadoop, australia, r, melbourne'
Topic 4
'university, research, canberra, mining, postdoctoral, position, seminar'
Topic 5
'r, data, mining, slides, rdatamining, series, group'
Topic 6
'example, us, r, knowledge, detection, outlier, thanks'
Topic 7
'mining, august, ausdm, poll, submission, cfp, processing'
Topic 8
'r, free, online, tutorial, course, package, mining'

In [71]:

```
library(data.table)
topics <- topics(lda) # 1st topic identified for every documen
t (tweet)
topics <- data.frame(date=as.IDate(tweets.df$created), topic=t
opics)
ggplot(topics, aes(date, fill = term[topic])) +
  geom_density(position = "stack")
```



Sentiment Analysis

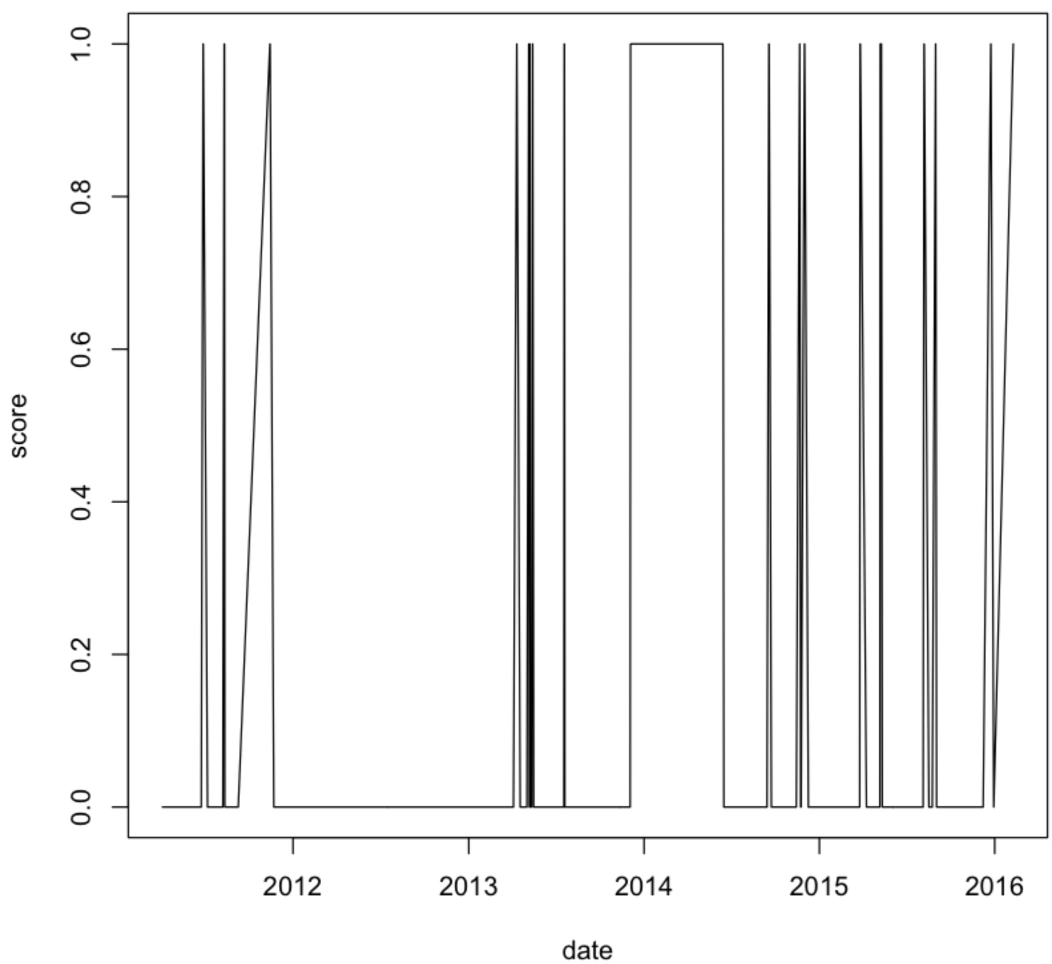
In [72]:

```
# different way to install a package
#require(devtools)
#install_github("sentiment140", "okugami79")

library(sentiment)

sentiments <- sentiment(tweets.df$text)
table(sentiments$polarity)
```

```
Loading required package: RCurl
Loading required package: bitops
Loading required package: rjson
Loading required package: plyr

Attaching package: ‘plyr’

The following object is masked from ‘package:grap
h’:

    join

The following object is masked from ‘package:twitte
R’:

    id


 neutral positive 
     428       20 
```

In [73]:

```
# sentiment plot
sentiments$score <- 0
sentiments$score[sentiments$polarity == "positive"] <- 1
sentiments$score[sentiments$polarity == "negative"] <- -1
sentiments$date <- as.IDate(tweets.df$created)
result <- aggregate(score ~ date, data = sentiments, sum)
plot(result, type = "l")
```

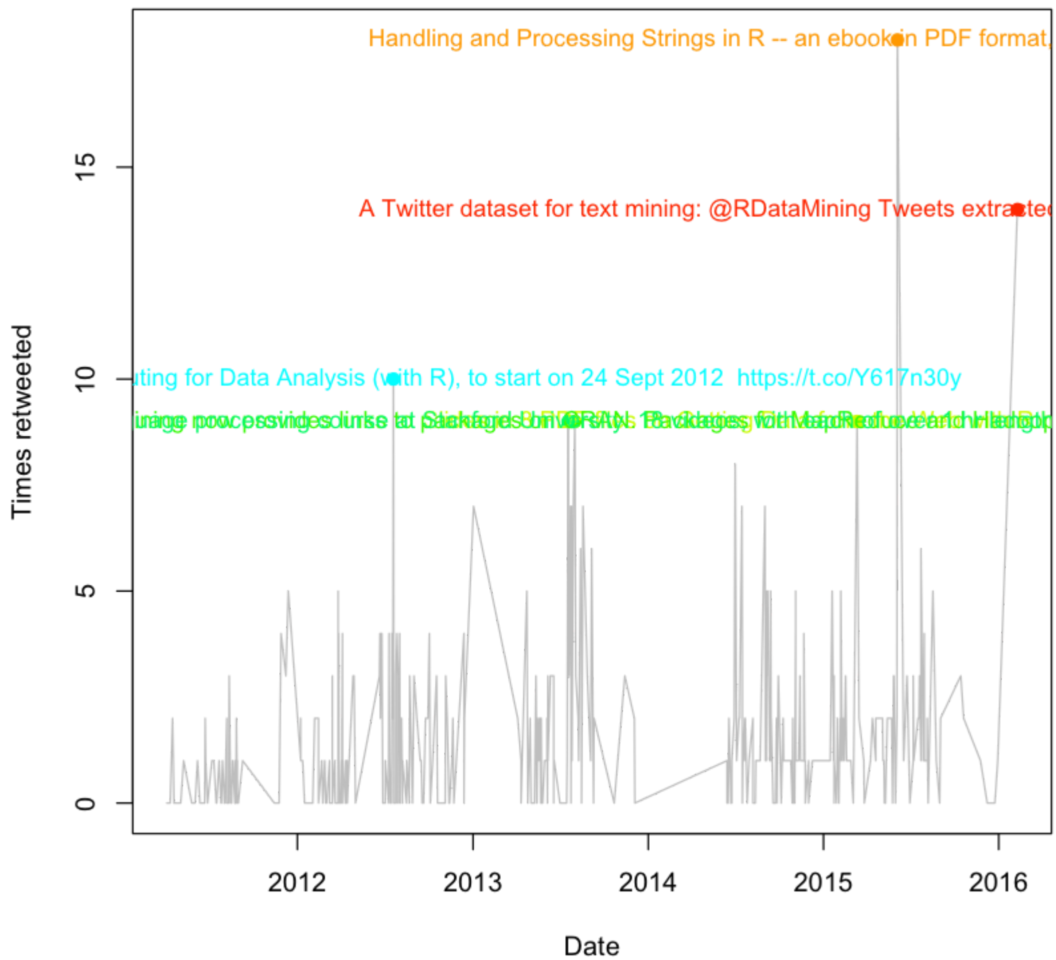


Top retweetet tweets

In [75]:

```
# select top retweeted tweets
table(tweets.df$retweetCount)
selected <- which(tweets.df$retweetCount >= 9)
# plot them
dates <- strptime(tweets.df$created, format="%Y-%m-%d")
plot(x=dates, y=tweets.df$retweetCount, type="l", col="grey",
     xlab="Date", ylab="Times retweeted")
colors <- rainbow(10)[1:length(selected)]
points(dates[selected], tweets.df$retweetCount[selected],
       pch=19, col=colors)
text(dates[selected], tweets.df$retweetCount[selected],
     tweets.df$text[selected], col=colors, cex=.9)
```

```
  0   1   2   3   4   5   6   7   8   9  10  14  18 
173 116  70  39  22  11   3   7   1   3   1   1   1 
```



Many more things that one wants to explore with Twitter data

e.g. Retrieve User Info and Followers

This Tutorial is based on: Yanchang Zhao [http://www.rdatamining.com/docs/twitter-analysis-with-r (http://www.rdatamining.com/docs/twitter-analysis-with-r)](http://www.rdatamining.com/docs/twitter-analysis-with-r)