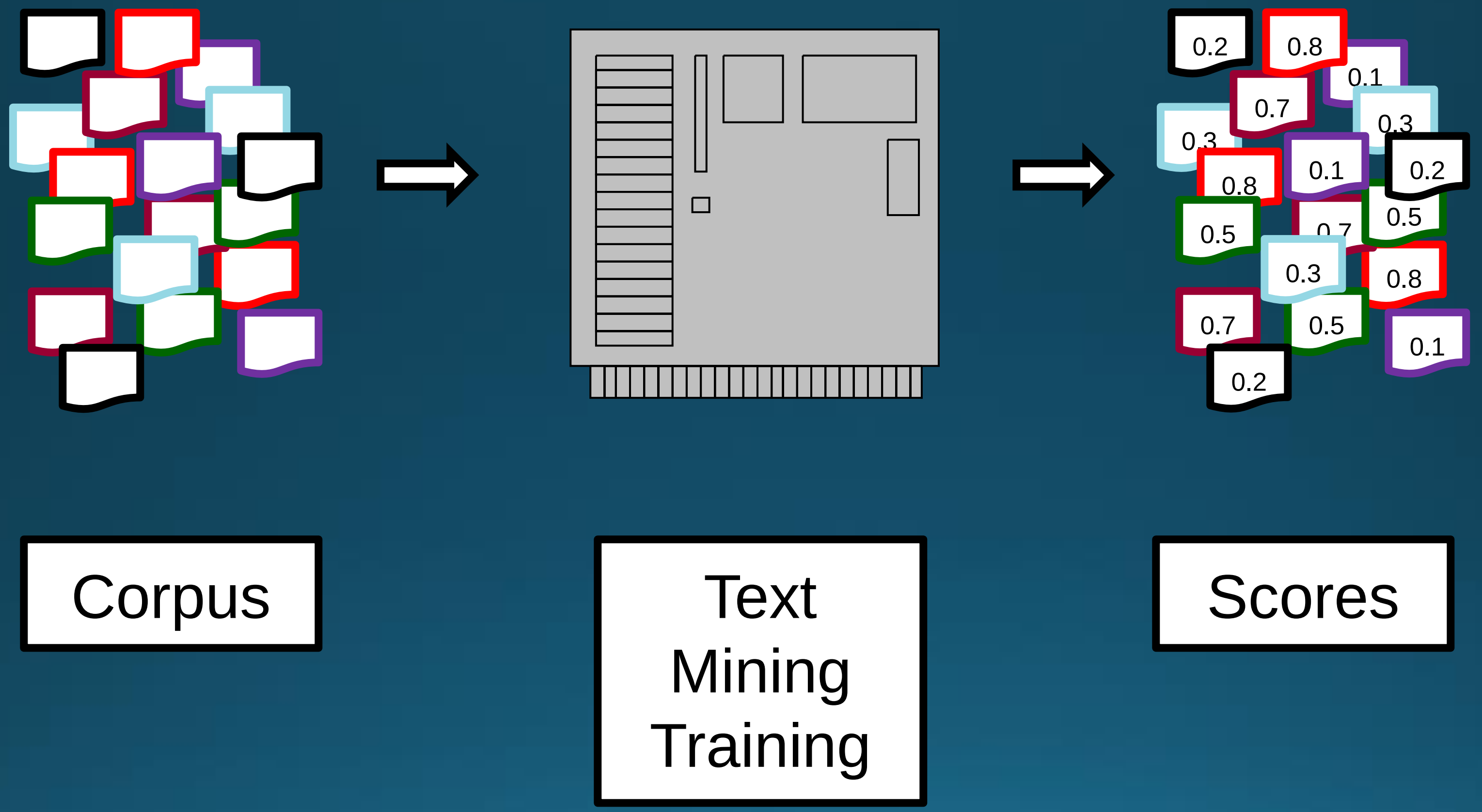
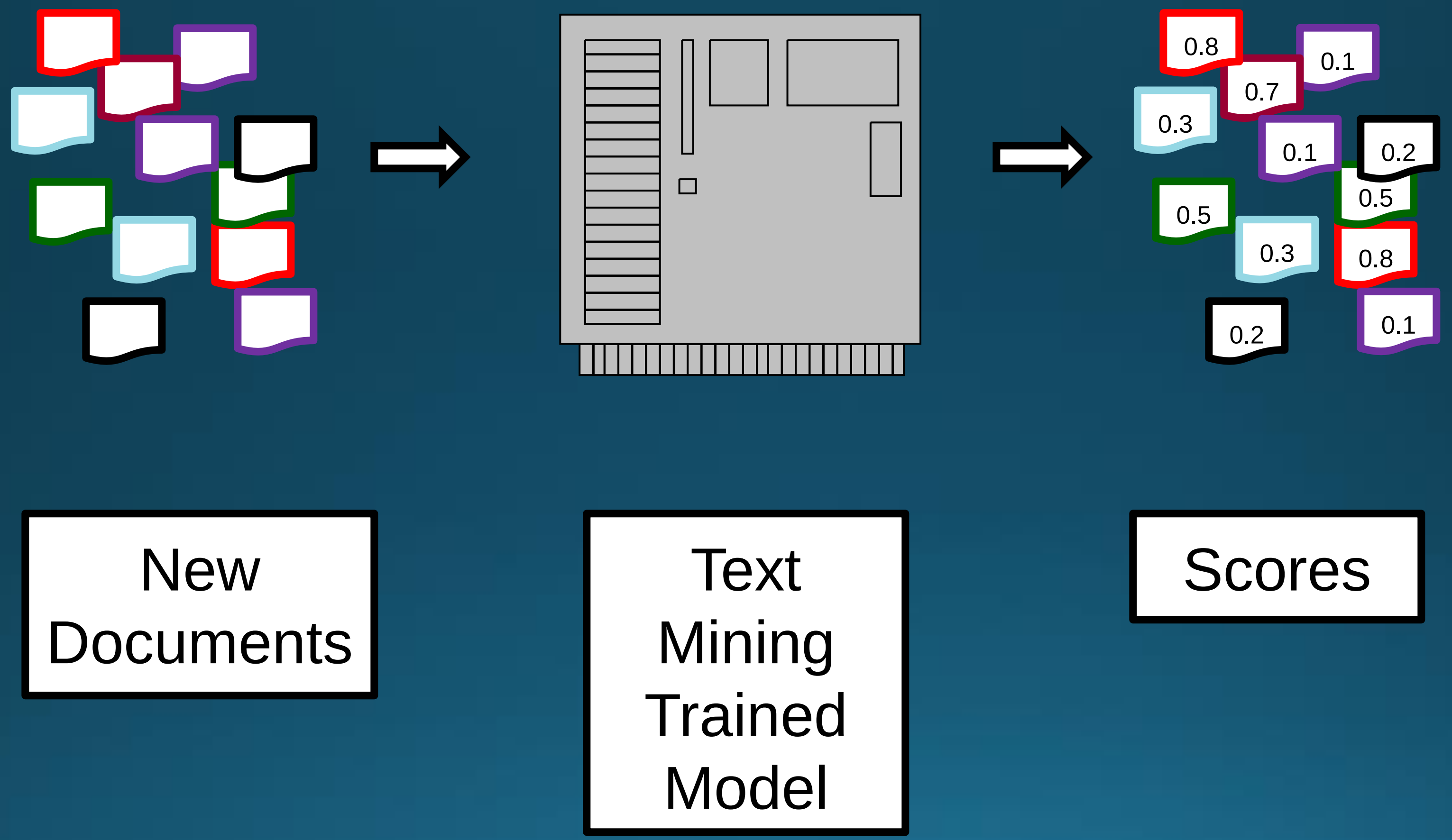
Process: Predictive Analysis

Text Mining and Data Mining

Columbia University
School of Professional Studies



Text Mining and Data Mining



Steps in Process

- Get text
- Prepare text
- Tokenization
- Dimensionality Reduction
- Weighting
- Predictive modelling with textual features

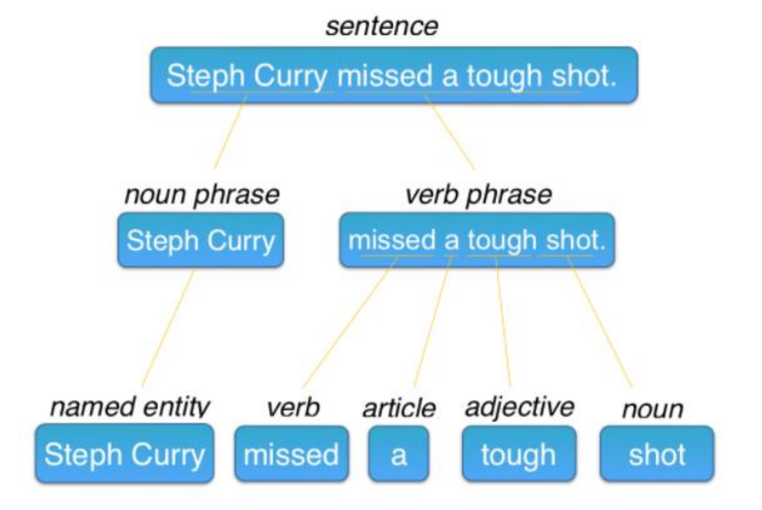
Get Text

- Manually Copy
- Pull from databases
- API (Application Programming Interface)
- Scrape

Prepare Text

- Bag of Words Approach
  - create corpus
  - lower case
  - remove punctuation
  - remove brackets
  - remove numbers
  - replace numbers, replace contraction, replace abbreviation, replace symbol
  - remove words, stop words
  - strip white space
  - stem document

- Semantic Parsing
  - Identify patterns (e.g., zip code, address, sentences)
  - Identify and group synonyms
  - Lexical diversity
  - Readability



Source: datacamp.com

Tokenization

- Process of breaking a stream of text, a character sequence or a defined document unit, into phrases, words or other meaningful elements called tokens
  - One word token: Unigram
  - Two word token: Bigram
  - n word token: n-Gram
- PoS Tagging
  - Annotation of word with the right part-of-speech tag. Basic tags include noun, verb, adjective, number and proper noun
  - Use PoS tag dictionary
- Chunking
  - Dividing text into syntactically correlated words like noun groups and verb groups or their role in the sentence

Dimensionality Reduction

- Tokenization of text will inevitably create a large number of dimensions relative to sample size. Too many dimensions leads to overfitting and in extreme cases (n<p) a unique solution may not even be possible. This is the curse of dimensionality.

- Techniques for dimensionality reduction with textual data
  - Drop rare token. For e.g., drop tokens that appear in fewer than 1% documents.
  - Principal Components Analysis
  - Correspondence Analysis
  - Singular Value Decomposition

- Document-term matrix: This is a sparse matrix describing a collection of documents with one row for each document and one column for each term.

	love	sweet	satirical	great	fun	whimsical	romantic	laughing	recommend	several	happy	again	horror	sad	unsure	waste	not
Doc 1	1	1	1	1	1	1	1	1	1	1	1	1					
Doc 2	1			2								1	1	1			
Doc 3						1	1		3						1	1	1

Weighting

- Terms may be weighted based on
  - Term Frequency
    - the simplest choice is to use the raw count of a term in a document
  - Term frequency – inverse document frequency
    - the inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all documents.
  - SMART Weighting – specifies a weight for term frequency, a weight for document frequency and then a schema for normalization

Predictive Model with Textual Features

- Append textual features to dataset containing non-text features and outcome variable
- Fit predictive model

R Illustration – Predictive Analysis with Text

Predictive Analysis with Text

About the Data

This dataset is based on a set of Amazon review of video games downloaded from Prof. Julian McAuley's website. The original json dataset was parsed into a dataframe and cleaned. Since this dataset is very large, I have only included data from Aug 2013 to July 2014 which gives us a year of data to work with. Also we are only using data on three fields, id, review and review rating. Finally, three blank reviews were dropped from the data. The data we are going to use includes the following fields:

- id: A unique identifier for each review
- review: Text of review posted on Amazon
- review_rating: Each review on Amazon is rated by others using a five-star scale (presumably based on helpfulness of review)

Read Data

You must read the data before trying to run the code on your own machine. To read data use the following code after setting your working directory.

```
videogame = read.csv('video_game_reviews.csv',stringsAsFactors = F)
```

Prepare Data

Clean and Tokenize

There are many ways to quantify text. Here, we will use the bag of words approach where we first clean the data and then extract individual words, a process known as tokenization. Specifically, we will create a corpus, clean the text, and generate a matrix derived from term frequencies.

1. Create a corpus from the variable 'review'
2. Use tm_map to

a. transform text to lower case,
b. remove punctuation,
c. remove English stopwords using the following dictionary tm::stopwords('english)
d. remove urls
e. remove whitespace

3. Create a dictionary
4. Use tm_map to stem words
5. Create a DocumentTermMatrix

You can learn more about cleaning functions in tm in this vignette We are going to make use of library(tm) and a few other text analysis libraries, but you based on analysis goal, you may want to consider other available libraries.

Create a corpus

```
library(tm)
corpus = Corpus(VectorSource(videogame$review))
```

Examine Review 617 (now a document)

```
corpus[[617]]
```

```
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 717
```

Examine content of Review 617

```
corpus[[617]][1]
```

```
## $content
## [1] "Like scottrocket3 said, you're out of your damn mind if you buy this for $200.  This is the best Mario game that ever came out.  The other is Super Mario World for Super Nintendo, also available for Nintendo DS.  I can't say enough about this great game!!  I LOVED the Super Mario Brothers Super Show featuring wrestling great Captain Lou Albano who, unfortunately passed away recently.  He was a Christian, so I will see him again.  Don't know about Danny Wells who did Luigi.  This is worth every penny you spend on it.  Unless of course you spend $100+dollars on this.  Mario first got me hooked on mushrooms.  Since then, I eat them by the truckload!!  They're good for you & have vitamin D, the sunshine vitamin."
```

Clean text

- convert to lower case
- remove urls
- remove punctuation
- remove stopwords
- remove whitespace

Convert to lower case

```
corpus = tm_map(corpus,FUN = content_transformer(tolower))
corpus[[617]][1]
```

```
## $content
## [1] "like scottrocket3 said, you're out of your damn mind if you buy this for $200.  this is the best mario game that ever came out.  the other is super mario world for super nintendo, also available for nintendo ds.  i can't say enough about this great game!!  i loved the super mario brothers super show featuring wrestling great captain lou albano who, unfortunately passed away recently.  he was a christian, so i will see him again.  don't know about danny wells who did luigi.  this is worth every penny you spend on it.  unless of course you spend $100+dollars on this.  mario first got me hooked on mushrooms.  since then, i eat them by the truckload!!  they're good for you & have vitamin d, the sunshine vitamin."
```

Remove urls

There are a few urls within the reviews. Use regular expressions to specify the pattern of a url. Replace urls with a blank.
For an introduction to regular expressions, see Chapter on Strings in R for Data Science or Regular Expressions Vignette on CRAN. For a quick search, use a RegEx Cheatsheet

Lets examine Review 4607 (instead of 617) since it has a url. Review 617 does not have a url to remove, but don't worry we will get back to 617 after this illustration.

```
corpus[[4607]][1]
```

```
## $content
## [1] "since ps2's and ps3's use the exact same composite or component cable, i figured this would wor
k........ guess i was wrong. for more info read this review, not typing it all again:http://www.amazon.com/rev
iew/r11mcq8esc5a3j/ref=cm_cr_rev_detup_redir?_encoding=utf8&asin;=b0017o5k0i&cdforum;=fx2rjz1hujht04o&cdpage;=
1&cdthread;=tx158cf6u8rxsnc&newcontentid;=mx1sf4qcpz70qcn&newcontentnum;=3&store;=pc#mx1dx5ei2d5fv5nedit!!: lo
l ok changing my review...... maybe the red video cord wasn't plugged in all the way or some weird crap, cause
i unplugged the cords one by one and plugged them back in and when i hit the red one and plugged it in agai
n..... poof i had color. i don't know..... but holy hell what a difference on my sony bravia. in final fantasy
12, things were so fuzzy on composite cable (the ps2's official composite cable, red/white/yellow) it looked u
gly as hell. with this? it's as clear as it is on the crt tv we still have in the living room....  but i did g
o to the ingame options and turn on flicker filter (keeps it from getting fuzzy when you/the camera move)then
on my tv's settings turn sharpness practically to max. but i couldn't even get anywhere near this good of a pi
cture with composite. in fact, i'm going to gamestop right now and getting a used ps3 (since they're the same
cords) component cable so i'll have this one as a spare. loli definitely recommend getting a component cable f
or ps2's on newer tv's, if not this one then another."
```

Match pattern and replace url with blank space.

```
corpus = tm_map(corpus,
                FUN = content_transformer(function(x)gsub(pattern = 'http[[:alnum:][:punct:]]*',
                                                           replacement = ' ',x = x)))
corpus[[4607]][1]
```

```
## $content
## [1] "since ps2's and ps3's use the exact same composite or component cable, i figured this would wor
k........ guess i was wrong. for more info read this review, not typing it all again:  lol ok changing m
y review...... maybe the red video cord wasn't plugged in all the way or some weird crap, cause i unplug
ged the cords one by one and plugged them back in and when i hit the red one and plugged it in agai
n..... poof i had color. i don't know..... but holy hell what a difference on my sony bravia. in final f
antasy 12, things were so fuzzy on composite cable (the ps2's official composite cable, red/white/yello
w) it looked ugly as hell. with this? it's as clear as it is on the crt tv we still have in the living r
oom....  but i did go to the ingame options and turn on flicker filter (keeps it from getting fuzzy when
you/the camera move)then on my tv's settings turn sharpness practically to max. but i couldn't even get
anywhere near this good of a picture with composite. in fact, i'm going to gamestop right now and gettin
g a used ps3 (since they're the same cords) component cable so i'll have this one as a spare. loli defin
itely recommend getting a component cable for ps2's on newer tv's, if not this one then another."
```

Console | Terminal | Jobs

```
> ?regex
>
```

Files | Plots | Packages | Help | Viewer

R: Regular Expressions as used in R | Find in Topic

regex {base} R Documentation

Regular Expressions as used in R

Description

This help page documents the regular expression patterns supported by `grep` and related functions `grepl`, `regexpr`, `gregexpr`, `sub` and `gsub`, as well as by `strsplit`.

A regular expression may be followed by one of several repetition quantifiers:

`?`

The preceding item is optional and will be matched at most once.

`*`

The preceding item will be matched zero or more times.

`+`

The preceding item will be matched one or more times.

`{n}`

The preceding item is matched exactly `n` times.

`[:alnum:]`

Alphanumeric characters: `[:alpha:]` and `[:digit:]`.

`[:alpha:]`

Alphabetic characters: `[:lower:]` and `[:upper:]`.

`[:blank:]`

Blank characters: space and tab, and possibly other locale-dependent characters such as non-breaking space.

`[:cntrl:]`

Control characters. In ASCII, these characters have octal codes 000 through 037, and 177 (`DEL`). In another character set, these are the equivalent characters, if any.

`[:digit:]`

Digits: `0 1 2 3 4 5 6 7 8 9`.

`[:graph:]`

Graphical characters: `[:alnum:]` and `[:punct:]`.

`[:lower:]`

Lower-case letters in the current locale.

`[:print:]`

Printable characters: `[:alnum:]`, `[:punct:]` and space.

`[:punct:]`

Punctuation characters:
`! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~`.

`[:space:]`

Space characters: tab, newline, vertical tab, form feed, carriage return, space and possibly other locale-dependent characters.

`[:upper:]`

Upper-case letters in the current locale.

`[:xdigit:]`

Hexadecimal digits:
`0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f`.

Remove punctuation

```
corpus = tm_map(corpus,FUN = removePunctuation)
corpus[[617]][1]
```

```
## $content
## [1] "like scottrocket3 said youre out of your damn mind if you buy this for 200  this is the best mario gam
e that ever came out  the other is super mario world for super nintendo also available for nintendo ds  i cant
say enough about this great game  i loved the super mario brothers super show featuring wrestling great captai
n lou albano who unfortunately passed away recently  he was a christian so i will see him again  dont know abo
ut danny wells who did luigi  this is worth every penny you spend on it  unless of course you spend 100dollars
on this  mario first got me hooked on mushrooms  since then i eat them by the truckload  theyre good for you
have vitamin d the sunshine vitamin"
```

Remove stopwords

```
corpus = tm_map(corpus,FUN = removeWords,c(stopwords('english')))
corpus[[617]][1]
```

```
## $content
## [1] "like scottrocket3 said youre    damn mind    buy    200     best mario game   ever came      super mario
world  super nintendo also available  nintendo ds    cant say enough    great game    loved  super mario brothers
super show featuring wrestling great captain lou albano  unfortunately passed away recently     christian   wi
ll see    dont know  danny wells    luigi    worth every penny  spend    unless  course  spend 100dollars    ma
rio first got  hooked  mushrooms  since    eat    truckload  theyre good     vitamin d  sunshine vitamin"
```

Strip whitespace

```
corpus = tm_map(corpus,FUN = stripWhitespace)
corpus[[617]][1]
```

```
## $content
## [1] "like scottrocket3 said youre damn mind buy 200 best mario game ever came super mario world super ninte
ndo also available nintendo ds cant say enough great game loved super mario brothers super show featuring wres
tling great captain lou albano unfortunately passed away recently christian will see dont know danny wells lui
gi worth every penny spend unless course spend 100dollars mario first got hooked mushrooms since eat truckload
theyre good vitamin d sunshine vitamin"
```

- computational, computers, and computation stem to "comput". However, "comput" is not a real word.
- Reconstruct "comput" into a recognizable word

Create a dictionary

Stemming chops off the end of words but the manner in which it does so may render the words meaningless. Here we are creating a dictionary of all words. Once, similar words have been grouped together, the part that was chopped off will be reintegrated using the dictionary created here.

```
dict = findFreqTerms(DocumentTermMatrix(Corpus(VectorSource(videogame$review))),
                     lowfreq = 0)
dict_corpus = Corpus(VectorSource(dict))
```

A numeric for the lower frequency bound.

Stem document

```
corpus = tm_map(corpus,FUN = stemDocument)
corpus[[617]][1]
```

```
## $content
## [1] "like scottrocket3 said your damn mind buy 200 best mario game ever came super mario
world super nintendo also avail nintendo ds cant say enough great game love super mario bro
ther super show featur wrestl great captain lou albano unfortun pass away recent christian
will see dont know danni well luigi worth everi penni spend unless cours spend 100dollar ma
rio first got hook mushroom sinc eat truckload theyr good vitamin d sunshin vitamin"
```

Create a document term matrix (tokenize)

COLUMBIA UNIVERSITY
School of Professional Studies

Note, here we are going with the default arguments which include using Term Frequency weighting for each term.

```
dtm = DocumentTermMatrix(corpus)
dtm
```

```
## <<DocumentTermMatrix (documents: 26652, terms: 65864)>>
## Non-/sparse entries: 1235636/1754171692
## Sparsity           : 100%
## Maximal term length: 115
## Weighting          : term frequency (tf)
```

Non-Sparse entries	1235636	
Sparse entries	1754171692	
Sparsity	sparse enties/ (non-sparse entries+ sparse entires)	0.999

Each review is represented as a document in the document term matrix. Let us see how many times the word "game" appears in this review.

```
inspect(dtm[617,])
```

Inspect document/review 617 of the document term matrix.

```
## <<DocumentTermMatrix (documents: 1, terms: 65864)>>
## Non-/sparse entries: 60/65804
## Sparsity           : 100%
## Maximal term length: 115
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  game good got great mario nintendo said spend super vitamin
##   617    2    1   1     2     4        2    1     2     4       2
```

Non-Sparse entries	60	
Sparse entries	65804	
Sparsity	sparse enties/ (non-sparse entries+ sparse entires)	1.000

```r
inspect(dtm[617,'game'])
```

```
## <<DocumentTermMatrix (documents: 1, terms: 1)>>
## Non-/sparse entries: 1/0
## Sparsity           : 0%
## Maximal term length: 4
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  game
##   617    2
```

- How many times does the word 'game' appear in document/review 617 of the document term matrix?

The document term matrix contains a column for each term generated from tokenizing the corpus.

```r
dim(dtm)
```

```
## [1] 26652 65864
```

Inspect terms 24001 to 24010 for reviews 611 to 620. This sparsity, few 1s and mostly 0s is the norm, not the exception.

```
inspect(dtm[611:620,24001:24010])
```

```
## <<DocumentTermMatrix (documents: 10, terms: 10)>>
## Non-/sparse entries: 0/100
## Sparsity           : 100%
## Maximal term length: 13
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  venezuelan veteranlevel dogthorough friendsi 2015after bc2bf3bf4 cod1uo
##   611          0            0           0        0         0         0      0
##   612          0            0           0        0         0         0      0
##   613          0            0           0        0         0         0      0
##   614          0            0           0        0         0         0      0
##   615          0            0           0        0         0         0      0
##   616          0            0           0        0         0         0      0
##   617          0            0           0        0         0         0      0
##   618          0            0           0        0         0         0      0
##   619          0            0           0        0         0         0      0
##   620          0            0           0        0         0         0      0
##      Terms
## Docs  disclaimers1 out2 performanceit
##   611            0    0             0
##   612            0    0             0
##   613            0    0             0
##   614            0    0             0
##   615            0    0             0
##   616            0    0             0
##   617            0    0             0
##   618            0    0             0
##   619            0    0             0
##   620            0    0             0
```

Remove Sparse Terms

The document term matrix contains a very large number of tokens. Frequently with text analysis one is faced with the curse of dimensionality where there are more variables than observations. A simple way to address this is to remove infrequently occurring terms. Here we will remove terms that appear in fewer than 5% of the reviews. In other words, we will retain all terms that appear in 5% or more reviews. Thereafter we convert the document-term-matrix to a data frame and address any problems with column names as our column names will now be the tokens (i.e., words in the texts).

```
xdtm = removeSparseTerms(dtm,sparse = 0.95)
xdtm
```

only keeping terms that appear in at least 5% of documents

```
## <<DocumentTermMatrix (documents: 26652, terms: 158)>>
## Non-/sparse entries: 447491/3763525
## Sparsity           : 89%
## Maximal term length: 9
## Weighting          : term frequency (tf)
```

terms remain after removing sparse terms

Complete Stems

We are going to use the dictionary created earlier to complete the stems. This should make the terms more meaningful. Since the terms will now serve as column names for a dataframe, we need to ensure the terms comply with variable naming conventions in R.

```
xdtm = as.data.frame(as.matrix(xdtm))
colnames(xdtm) = stemCompletion(x = colnames(xdtm),
                                dictionary = dict_corpus,
                                type='prevalent')
colnames(xdtm) = make.names(colnames(xdtm))
```

- computational, computers, and computation stem to “comput”. However, “comput” is not a real word.
- Reconstruct “comput” into a recognizable word
- prevalent- Default in stemCompletion(). Takes the most frequent match as completion.

- Let's take a look at the complete stemmed data frame, which term appears most frequently?

COLUMBIA UNIVERSITY
School of Professional Studies

Browse tokens

Next, we evaluate the frequency of the tokens as these frequencies will be used to weight the tokens. A word that occurs more frequently gets weighted more than one that appears less frequently.

```
sort(colSums(xdtm),decreasing = T)
```

```
##        game      play      like       one       get
##       69555     22510     17771     15323     14463
##         can      just      time     great      will
##       14460     13184     10937     10333      9745
##         use      good   reallisc       fun   control
##        9711      9327      9304      8405      8267
##        much      make      love      dont      even
##        7424      7352      6791      6533      6447
##        look      also    storie     first      well
##        6427      6396      6315      6118      6097
##        work       new character     thing      want
##        5857      5844      5807      5737      5514
##       still   graphic       lot      feel    better
##        5413      5350      5299      5161      5000
##         way       buy     level       say      xbox
##        4904      4786      4340      4329      4254
##        take      best       now     enjoy      need
```

The term 'game' appears most frequently

Document Term Matrix - tfidf

Now, we are going to consider another document term matrix, this time using Term Frequency - Inverse Document Frequency Weighting. Since the code is similar to the previous (term frequency) document term matrix, we will run the code in a single block.

```
dtm_tfidf = DocumentTermMatrix(x=corpus,
                               control = list(weighting=function(x) weightTfIdf(x,normalize=F)))
xdtm_tfidf = removeSparseTerms(dtm_tfidf,sparse = 0.95)
xdtm_tfidf = as.data.frame(as.matrix(xdtm_tfidf))
colnames(xdtm_tfidf) = stemCompletion(x = colnames(xdtm_tfidf),
                                      dictionary = dict_corpus,
                                      type='prevalent')
colnames(xdtm_tfidf) = make.names(colnames(xdtm_tfidf))
sort(colSums(xdtm_tfidf),decreasing = T)
```

```
##        game       can      play      like       one
##   32408.595 28486.212 27180.275 27026.206 25401.440
##         get      just      will      time       use
##   25055.553 24011.027 22689.043 22586.054 22509.923
##     control  reallisc     great      good      make
##   21846.471 20560.469 18995.677 18839.042 18510.135
```

Document Term Matrix: Term Frequency vs. Term Frequency Inverse Document Frequency

```
xdtm[611:620,41:50]
```

	complete	consola	control	end	even	ever	fan	gameplay
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
611	0	0	0	0	0	0	0	0
612	0	0	1	0	1	0	0	0
613	0	0	0	0	0	0	0	0
614	0	0	0	0	0	0	0	0
615	0	0	0	0	0	0	0	0
616	0	0	0	0	0	0	0	0
617	0	0	0	0	0	1	0	0
618	0	0	2	0	0	0	0	0
619	0	0	0	0	1	0	1	0
620	0	0	0	0	0	0	0	0

1-10 of 10 rows | 1-10 of 11 columns

Columbia University
School of Professional Studies

```
xdtm_tfidf[611:620,41:50]
```

	complete	consola	control	e...	even	ever	fan	gameplay	give
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
611	0	0	0.000000	0	0.000000	0.000000	0.000000	0	3.351569
612	0	0	2.642612	0	2.658587	0.000000	0.000000	0	0.000000
613	0	0	0.000000	0	0.000000	0.000000	0.000000	0	0.000000
614	0	0	0.000000	0	0.000000	0.000000	0.000000	0	0.000000
615	0	0	0.000000	0	0.000000	0.000000	0.000000	0	0.000000
616	0	0	0.000000	0	0.000000	0.000000	0.000000	0	0.000000
617	0	0	0.000000	0	0.000000	3.776402	0.000000	0	0.000000
618	0	0	5.285223	0	0.000000	0.000000	0.000000	0	6.703139
619	0	0	0.000000	0	2.658587	0.000000	3.498608	0	0.000000
620	0	0	0.000000	0	0.000000	0.000000	0.000000	0	0.000000

1-10 of 10 rows | 1-10 of 11 columns

The following horizontal bar chart contrasts the weights of term frequency and term frequency inverse document frequency weighting for the top 20 terms. Most noteworthy is the heavy weighting of the term “game”. Term Frequency assigns it a heavy weight because “game” is the most frequently occuring term. On the other hand, the word “game” appears in most of the reviews thus has little diagnostic value. Accordingly, Term Frequency - Inverse Document Frequency assigns “game” a much lower weight.

```
library(tidyr); library(dplyr); library(ggplot2); library(ggthemes)
data.frame(term = colnames(xdtm),tf = colMeans(xdtm), tfidf = colMeans(xdtm_tfidf))%>%
  arrange(desc(tf))%>%
  top_n(20)%>%
  gather(key=weighting_method,value=weight,2:3)%>%
  ggplot(aes(x=term,y=weight,fill=weighting_method))+
  geom_col(position='dodge')+
  coord_flip()+
  theme_economist()
```

```
+     gather(key=weighting_method,value=weight,2:3)
Selecting by tfidf
        term weighting_method    weight
1       game               tf 2.6097479
2       play               tf 0.8445895
3       like               tf 0.6667792
4        one               tf 0.5749287
5        get               tf 0.5426610
6        can               tf 0.5425484
7       just               tf 0.4946721
8       time               tf 0.4103632
9      great               tf 0.3877007
10      will               tf 0.3656386
11       use               tf 0.3643629
12      good               tf 0.3499550
13  reallisc               tf 0.3490920
14       fun               tf 0.3153609
15   control               tf 0.3101831
16      much               tf 0.2785532
17      make               tf 0.2758517
18      also               tf 0.2399820
19    storie               tf 0.2369428
20 character               tf 0.2178823
21      game            tfidf 1.2159911
22      play            tfidf 1.0198212
23      like            tfidf 1.0140405
24       one            tfidf 0.9530782
```

Add review_rating back to dataframe of features

```
videogame_data = cbind(review_rating = videogame$review_rating,xdtm)
videogame_data_tfidf = cbind(review_rating = videogame$review_rating,xdtm_tfidf)
```

Which is the second (2nd) most frequently occuring word among reviews with a rating of 5 when using:
* Document Term Matrix: Term Frequency
* Document Term Matrix: Term Frequency Inverse Document Frequency

```
sort(colSums(videogame_data[videogame_data$review_rating==5,-videogame_data$review_rating]),decreasing = T)
```

```
##          play         like          one          get        great
##         12213         8216         8063         6703         6680
##          just         time          use         love         will
##          5808         5107         4974         4899         4892
```

```
sort(colSums(videogame_data_tfidf[videogame_data$review_rating==5,-videogame_data_tfidf$review_rating]),decreasing = T)
```

```
##       play       one      like     great       get
##  14746.899 13366.300 12494.925 12280.182 11612.209
##        use      will      love   control      just
##  11529.642 11389.923 11383.617 10694.650 10577.673
```

Predictive Models (using TF features)

Split Data (TF)

```
set.seed(617)
split = sample(1:nrow(videogame_data),size = 0.7*nrow(videogame_data))
train = videogame_data[split,]
test = videogame_data[-split,]
```

- Split the dataset containing review rating and term frequencies into a train and test samples.
- Use sample() to create a train sample with 70% of the data and test sample with the remaining 30%. Use a seed of 617.

CART

- Use a CART model to predict review_rating using all other variables, i.e., term frequencies.


```
library(rpart); library(rpart.plot)
set.seed(100)
tree = rpart(review_rating~.,train)
rpart.plot(tree)
```

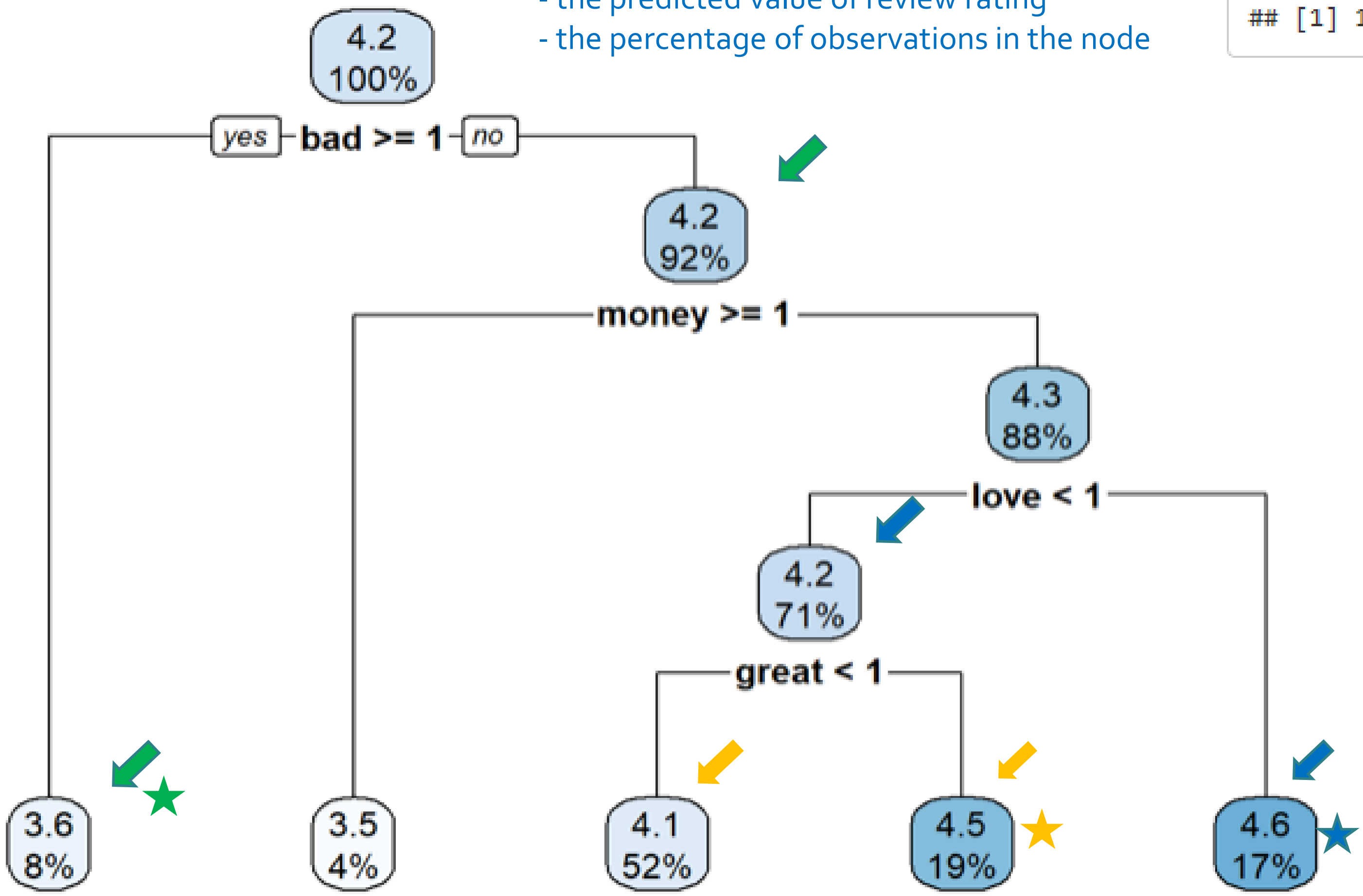
Predictions

```
pred_tree = predict(tree,newdata=test)
rmse_tree = sqrt(mean((pred_tree - test$review_rating)^2)); rmse_tree
```

```
## [1] 1.139024
```

rmse of the CART model on the test set

Each node shows
- the predicted value of review rating
- the percentage of observations in the node



- reviews that contain the term 'love' are rated higher than those that don't contain the term 'love'
- reviews that contain the term 'bad' are rated lower than those that don't contain the term 'bad'
- reviews that contain the term "great' are rated higher than those that don't contain the term 'great'

Regression

```r
reg = lm(review_rating~.,train)
summary(reg)
```

```
##
## Call:
## lm(formula = review_rating ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6334 -0.4454  0.4052  0.7585  4.3795
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.1324618  0.0118230 349.527  < 2e-16 ***
## can          0.0661146  0.0085388   7.743 1.02e-14 ***
## game        -0.0242193  0.0038849  -6.234 4.64e-10 ***
## good         0.0518814  0.0124763   4.158 3.22e-05 ***
## got         -0.0073809  0.0197083  -0.375 0.708031
## nother      -0.2601205  0.0291932  -8.910  < 2e-16 ***
## one          0.0261983  0.0096845   2.705 0.006833 **
## bad         -0.2871157  0.0222442 -12.907  < 2e-16 ***
## didnt       -0.0975079  0.0232077  -4.202 2.66e-05 ***
## dont        -0.0865969  0.0153148  -5.654 1.59e-08 ***
```

COLUMBIA UNIVERSITY
School of Professional Studies

- Is the most frequently occurring term predictive of review rating?

```
## review      -0.0044587  0.0197148  -0.226 0.821081
## part         0.0193071  0.0249176   0.775 0.438446
## ps4         -0.0293490  0.0157135  -1.868 0.061812 .
## doesnt      -0.1174874  0.0248685  -4.724 2.33e-06 ***
## money       -0.4659074  0.0308812 -15.087  < 2e-16 ***
## xbox        -0.0017449  0.0124898  -0.140 0.888893
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.094 on 18497 degrees of freedom
## Multiple R-squared:  0.1628, Adjusted R-squared:  0.1557
## F-statistic: 22.77 on 158 and 18497 DF,  p-value: < 2.2e-16
```

Predictions

```r
pred_reg = predict(reg, newdata=test)
rmse_reg = sqrt(mean((pred_reg-test$review_rating)^2)); rmse_reg
```

```
## [1] 1.093325
```

rmse of the linear regression model on the test set

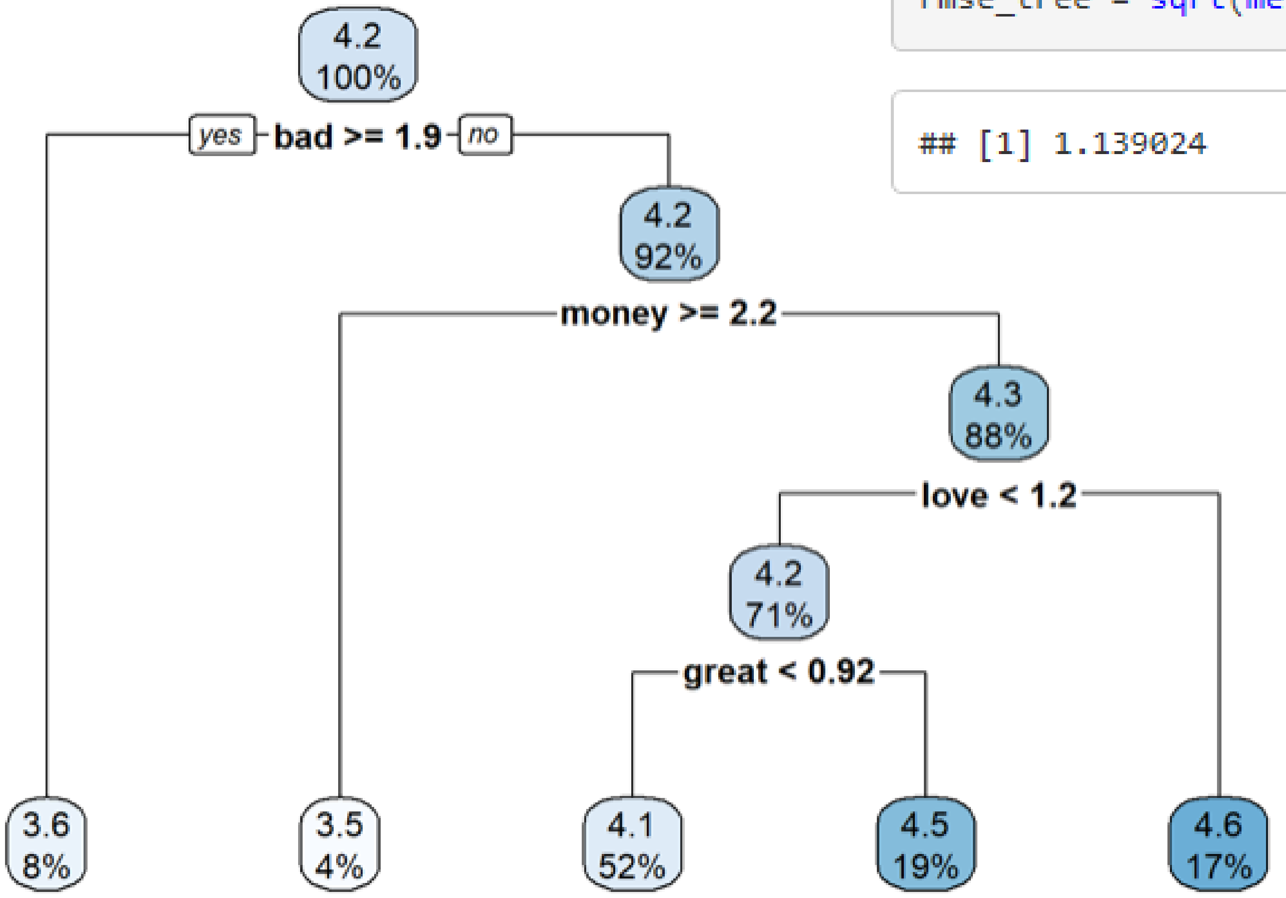
Predictive Models (using TF-IDF features)

Split Data (TF-IDF)

```
set.seed(617)
split = sample(1:nrow(videogame_data_tfidf),size = 0.7*nrow(videogame_data_tfidf))
train = videogame_data_tfidf[split,]
test = videogame_data_tfidf[-split,]
```

CART


```
library(rpart); library(rpart.plot)
set.seed(100)
tree = rpart(review_rating~.,train)
rpart.plot(tree)
```



Predictions

```
pred_tree = predict(tree,newdata=test)
rmse_tree = sqrt(mean((pred_tree - test$review_rating)^2)); rmse_tree
```

```
## [1] 1.139024
```

Regression

```
reg = lm(review_rating~.,train)
summary(reg)
```

```
##
## Call:
## lm(formula = review_rating ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6334 -0.4454  0.4052  0.7585  4.3795
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.1324618  0.0118230 349.527  < 2e-16 ***
## can          0.0335607  0.0043344   7.743 1.02e-14 ***
## game        -0.0519793  0.0083378  -6.234 4.64e-10 ***
## good         0.0256859  0.0061769   4.158 3.22e-05 ***
## got         -0.0024054  0.0064228  -0.375 0.708031
## nother      -0.0618289  0.0069390  -8.910  < 2e-16 ***
## one          0.0158037  0.0058420   2.705 0.006833 **
## bad         -0.0773806  0.0059950 -12.907  < 2e-16 ***
## didnt       -0.0265995  0.0063309  -4.202 2.66e-05 ***
## dont        -0.0337614  0.0059708  -5.654 1.59e-08 ***
```

```
## put         -0.0053235  0.0067168  -0.793 0.428042
## review      -0.0011911  0.0052665  -0.226 0.821081
## part         0.0048448  0.0062526   0.775 0.438446
## ps4         -0.0069744  0.0037341  -1.868 0.061812 .
## doesnt      -0.0315612  0.0066805  -4.724 2.33e-06 ***
## money       -0.1080005  0.0071585 -15.087  < 2e-16 ***
## xbox        -0.0004893  0.0035022  -0.140 0.888893
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.094 on 18497 degrees of freedom
## Multiple R-squared:  0.1628, Adjusted R-squared:  0.1557
## F-statistic: 22.77 on 158 and 18497 DF,  p-value: < 2.2e-16
```

Predictions

```
pred_reg = predict(reg, newdata=test)
rmse_reg = sqrt(mean((pred_reg-test$review_rating)^2)); rmse_reg
```

```
## [1] 1.093325
```

R Illustration - twitter

twitter

Gather Tweets

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

```
cat(paste('Gathered on ',month(ymd(Sys.Date()),label = T), ' ',
          day(ymd(Sys.Date())),',',year(ymd(Sys.Date())),sep = ''))
```

```
## Gathered on Feb 12,2022
```

search for the first 1000 tweets which contain the term JoeBiden.

```
library(rtweet)
tweets = search_tweets(q = 'JoeBiden',n = 1000, include_rts = F)
```

To export and save data use the following code after setting your working directory. To set your working directory, modify the following to set the file path for the folder where the data file resides. `setwd('c:/thatawesomeclass/')`

```
save_as_csv(tweets,'tweets.csv', prepend_ids =TRUE, na ="")
```

More information about using tokens :
https://cran.r-project.org/web/packages/rtweet/vignettes/auth.html

Review a few tweets

```
head(tweets)
```

user_id <chr>	status_id <chr>	created_at <dttm>
2240074220	1492402797429493762	2022-02-12 07:38:49
23336276	1492402778580430852	2022-02-12 07:38:44
36409600	1492402769587838978	2022-02-12 07:38:42
36409600	1492401872354914305	2022-02-12 07:35:08
36409600	1492401499435114500	2022-02-12 07:33:39
1478805790446080000	1492402750369710080	2022-02-12 07:38:37

6 rows | 1-3 of 90 columns

text <chr>
@hatejacktoo @LawyerOnSkis @JoeBiden When did they let you out of the psychiatric Institute. You should probably take your meds. Probably double up.
@Heavens_shoppe @JoeBiden That doesn't make sense considering there still cars continue education specialization in electrical vehicles it's not like the billions of gas cars are gonna disappear overnight anyway, maybe your boyfriend is just overqualified for everything
@Asensii20 @JoeBiden I'll just fucking make him King then
@Heavens_shoppe @JoeBiden Those must be the old throw away people that they wanted to kill off during the pandemic
@AlisonBoxxer @JoeBiden full of shit like Joe let's go Brandon
@BTNFINANCE good project @jack @JoeBiden @TheCryptoLark

6 rows | 5-5 of 90 columns

```r
tweets[1:10,c('screen_name','text')]
```

COLUMBIA UNIVERSITY
School of Professional Studies

screen_name <chr>
acquitted24
EdgarCorley
birdmanbob4
birdmanbob4
birdmanbob4
kimthanhsang
Stizmaster
ba110ba
ba110ba
ba110ba

1-10 of 10 rows | 1-1 of 2 columns

text <chr>
@hatejacktoo @LawyerOnSkis @JoeBiden When did they let you out of the psychiatric Institute. You should probably take your meds. Probably double up.
@Heavens_shoppe @JoeBiden That doesn't make sense considering there still cars continue education specialization in electrical vehicles it's not like the billions of gas cars are gonna disappear overnight anyway, maybe your boyfriend is just overqualified for everything
@Asensii20 @JoeBiden I'll just fucking make him King then
@Heavens_shoppe @JoeBiden Those must be the old throw away people that they wanted to kill off during the pandemic
@AlisonBoxxer @JoeBiden full of shit like Joe let's go Brandon
@BTNFINANCE good project @jack @JoeBiden @TheCryptoLark
@debbieditybaby @joeecollins3 @JoeBiden @RepMaxineWaters Remember better… because you're lying
@JoeBiden @ABlinken @thejointstaff @SenSchumer @SpeakerPelosi (1 of 2) it's important to work with President Zelensky in Kiev to move him ; other top brass to another safe location so when attacked by the Russia on Kiev does not allow them to take over the government. I hope it -
@JoeBiden @ABlinken @thejointstaff @SenSchumer @SpeakerPelosi (1 of 2) The US, as an ally at least, needs to show that we're not fools with Russia's potential invasion. We need to show Russia we will stand up for Ukraine & other eastern allies -
@JoeBiden @ABlinken @thejointstaff @SenSchumer @SpeakerPelosi (2 of 2) As an ally to Ukraine, we need to a) bring in the latest Patriot missile systems with our own crew to run them b) have F16s (w/ adv targeting sys that coord w/the F22 & wk w/Ukrainians, F22 & F35s in Poland

1-10 of 10 rows | 2-2 of 2 columns

Extract words from all tweets. Examine first fifty rows of data. Note the change in the data from wide format to a long/tall format.

```
library(tidytext); library(dplyr)

tweets_words = tweets%>%
  group_by(screen_name)%>%
  unnest_tokens(output = word,input = text)%>%
  ungroup()%>%
  mutate(row=1:n())
as.data.frame(tweets_words)[1:50,c('screen_name','word')]
```

	screen_name <chr>	word <chr>
1	acquitted24	hatejacktoo
2	acquitted24	lawyeronskis
3	acquitted24	joebiden
4	acquitted24	when
5	acquitted24	did
6	acquitted24	they
7	acquitted24	let
8	acquitted24	you
9	acquitted24	out
10	acquitted24	of

1-10 of 50 rows   Previous 1 2 3 4 5 Next

Binary Sentiment

There are a number of word lexicons that can be used to classify words as being positive or negative. The Bing lexicon categorizes words as being positive and negative.

```
as.data.frame(get_sentiments('bing'))[1:50,]
```

	word <chr>	sentiment <chr>
1	2-faces	negative
2	abnormal	negative
3	abolish	negative
4	abominable	negative
5	abominably	negative
6	abominate	negative
7	abomination	negative
8	abort	negative
9	aborted	negative
10	aborts	negative

1-10 of 50 rows   Previous 1 2 3 4 5 Next

COLUMBIA UNIVERSITY
School of Professional Studies

```
get_sentiments('bing')%>%
  group_by(sentiment)%>%
  count()%>%
  ungroup()
```

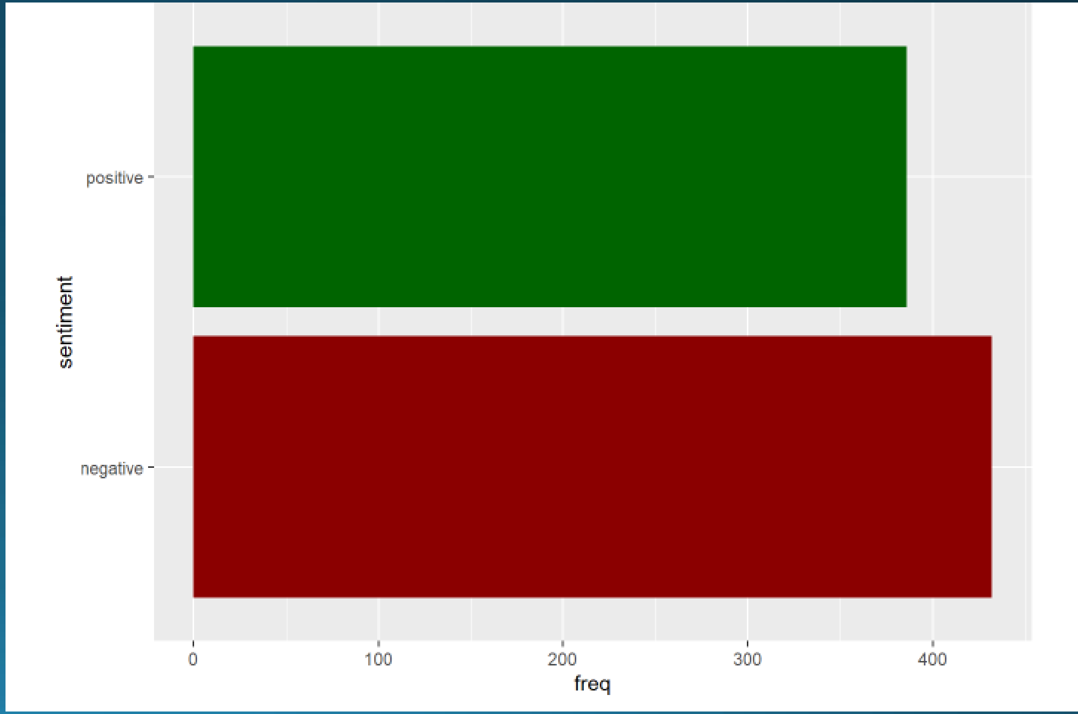
sentiment <chr>	n <int>
negative	4781
positive	2005

2 rows

Bing

Using the Bing Lexicon we can categorize all words in the tweets by valence. Here is a summary chart of positive and negative words

```
library(ggplot2)
tweets_words %>%
  inner_join(get_sentiments('bing'),by = 'word')%>%
  select('sentiment')%>%
  group_by(sentiment)%>%
  summarize(freq=n())%>%
  ungroup()%>%
  ggplot(aes(x=sentiment,y=freq))+geom_bar(position='dodge',
                                          stat='identity',fill=c('darkred','darkgreen'))+
  coord_flip()
```



Sentiment

Columbia University
School of Professional Studies

Another popular Lexicon is AFINN. This lexicon (available here) of words assigns scores words on the extent to which they are positive or negative. To make this code portable and seamless, the afinn lexicon has been placed on github from where it is being read in. Another alternative is to use get_sentiments('afinn') from library(tidytext) but it does not work in non-interactive mode

```
afinn = read.table('https://raw.githubusercontent.com/pseudorational/data/master/AFINN-111.txt',
                   header = F,sep = '\t',col.names = c('word','value'))
afinn[1:50,]
```

	word <fctr>	value <int>
1	abandon	-2
2	abandoned	-2
3	abandons	-2
4	abducted	-2
5	abduction	-2
6	abductions	-2
7	abhor	-3
8	abhorred	-3
9	abhorrent	-3
10	abhors	-3

1-10 of 50 rows Previous 1 2 3 4 5 Next

```
afinn %>%
  group_by(value)%>%
  count()
```

value <int>	n <int>
-5	4
-4	4
-3	43
-2	99
-1	49
1	33
2	68
3	35
4	3
5	1

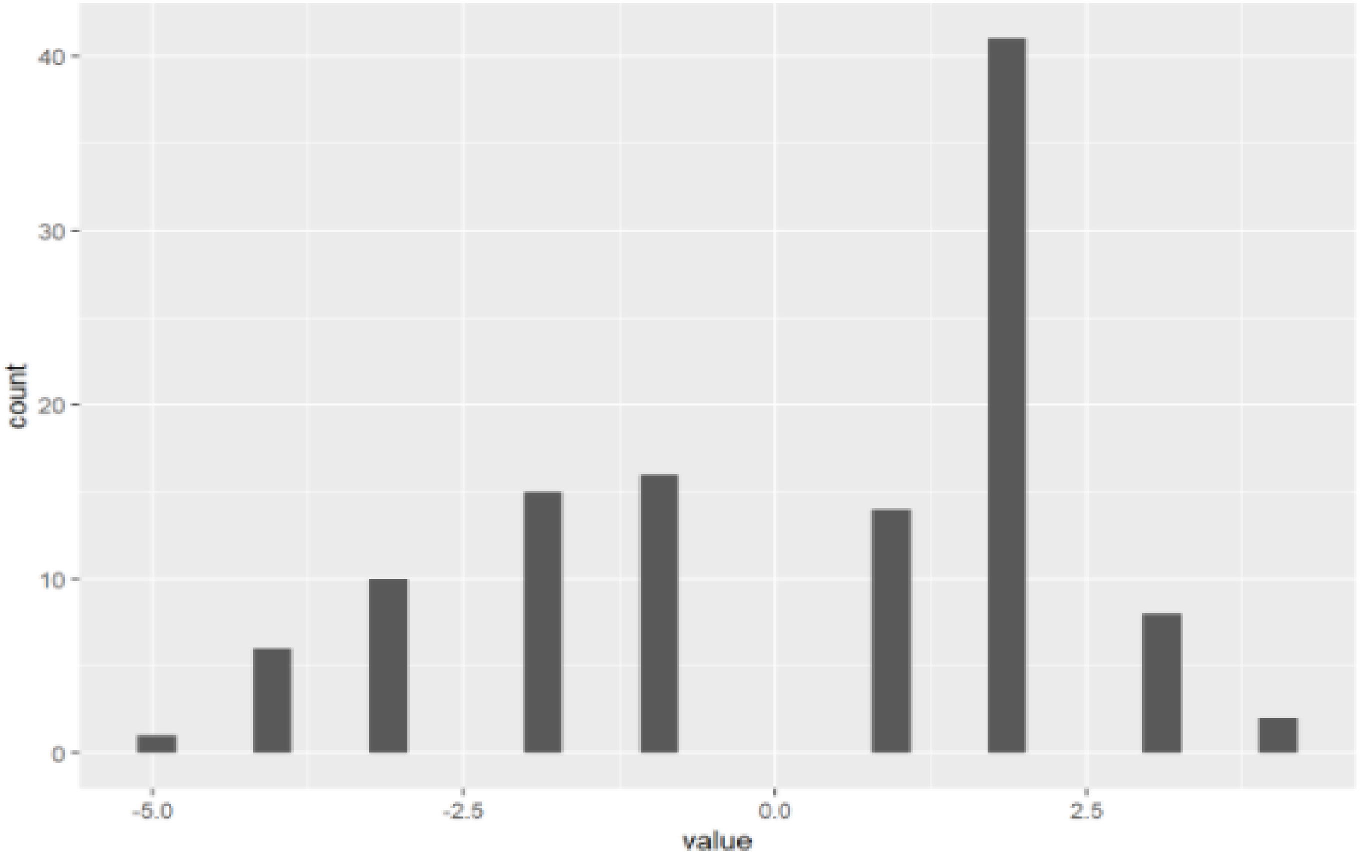
1-10 of 1... Previous 1 2 Next

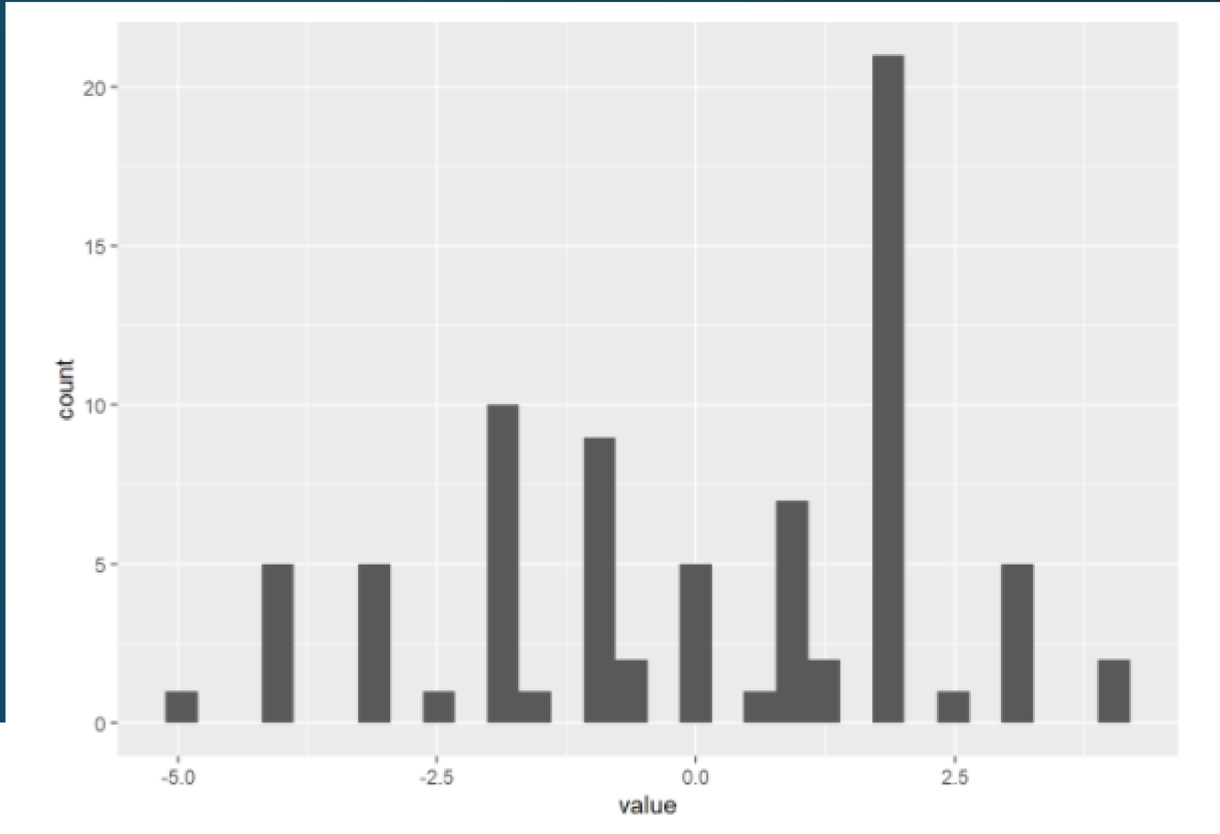
Scores all words

By applying the afinn lexicon to the words in the tweets, we can compute a score for each tweet. By averaging the scores of all tweets, one can obtain an overall sentiment score.

Here is a histogram of the scores of all words in the tweets.

```
tweets_words %>%
  inner_join(afinn,by = 'word')%>%
  select('value')%>%
  ggplot(aes(x=value))+geom_histogram()
```





Score each tweet

Here is a histogram of the scores of each tweet

Tweet(s) by screen name

```
tweets_words %>%
  left_join(afinn,by = 'word')%>%
  group_by(screen_name)%>%
  summarize(value = mean(value,na.rm=T))%>%
  ungroup()%>%
  select('screen_name','value')%>%
  ggplot(aes(x=value))+geom_histogram()
```

Sentiment Score

Sentiment score is the average of the sentiment of all tweets. First, we get the sentiment for each tweet

```
tweets_words %>%
  inner_join(afinn,by = 'word')%>%
  group_by(screen_name)%>%
  summarize(tweet_sentiment = mean(value,na.rm=T))%>%
  ungroup()
```

screen_name <chr>	tweet_sentiment <dbl>
AbdulGhafoor_Ar	-1.00
AlexWitzleben	0.50
aprilmorton73	-4.00
AZ_IBM_7090	1.00
ba110ba	0.00
BaaghiTV	2.00
balalogy	0.00
BBJAKK1	-2.00
BlingWendy	-3.00
chadsecor	2.00

1-10 of 78 rows Previous 1 2 3 4 ... 8 Next

```
tweets_words %>%
  inner_join(afinn,by = 'word')%>%
  group_by(screen_name)%>%
  summarize(tweet_sentiment = mean(value,na.rm=T))%>%
  ungroup()%>%
  summarize(Overall_Sentiment=mean(tweet_sentiment,na.rm=T))
```

Overall_Sentiment <dbl>
0.04038462

1 row

Emotions

A word may reflect more than just valence. The 'nrc' lexicon categorizes words by emotion. This lexicon which was previously a part of library(tidytext) was dropped from the package as of June 14, 2019. The lexicon was copied from its non-commercial use link and posted to github. This lexicon and a number of others can be found here but its free use is limited to non-commercial purposes. The following code will place the lexicon in a dataframe called nrc.

```
nrc = read.table(file = 'https://raw.githubusercontent.com/pseudorational/data/master/nrc_lexicon.txt',
                 header = F,col.names = c('word','sentiment','num'),sep = '\t');
                           nrc = nrc[nrc$num!=0,]; nrc$num = NULL
```

Summary of the number of words in the lexicon reflecting each emotion

```
nrc%>%
  group_by(sentiment)%>%
  count()
```

sentiment <chr>	n <int>
anger	1247
anticipation	839
disgust	1058
fear	1476
joy	689
negative	3324
positive	2312
sadness	1191
surprise	534
trust	1231

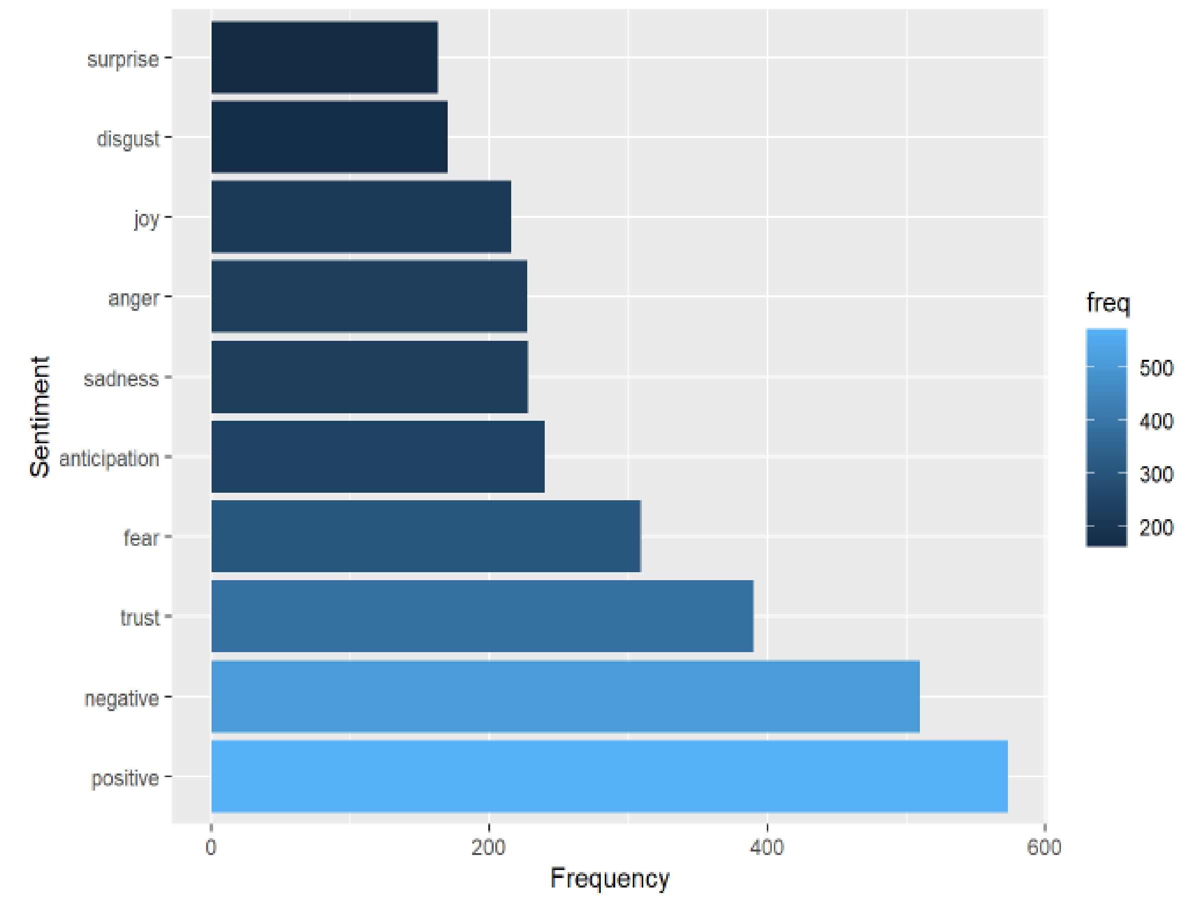
1-10 of 10 rows

Chart

Columbia University
School of Professional Studies

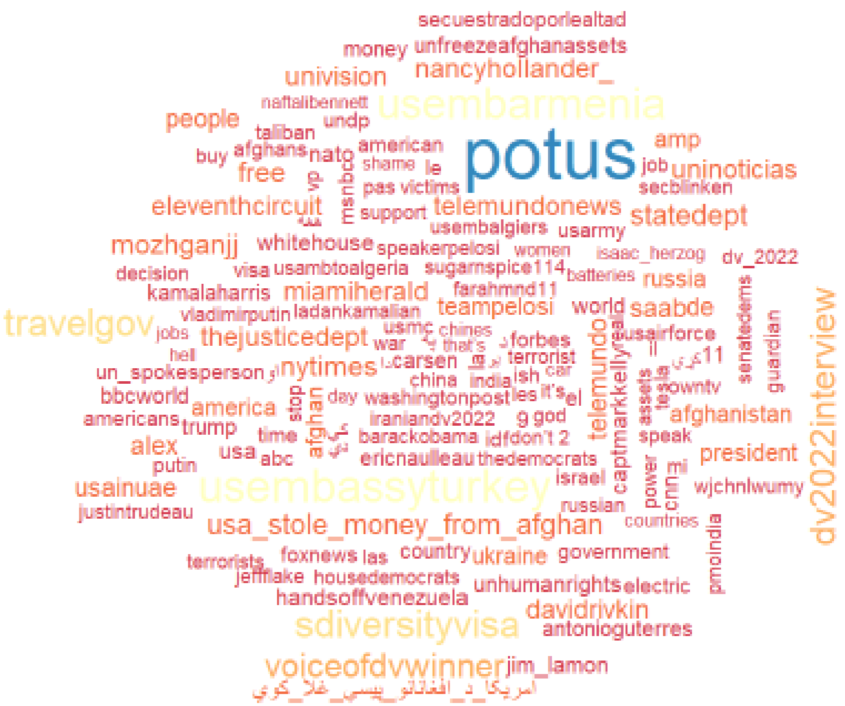
Chart of Words reflecting Emotion

```
library(RColorBrewer)
tweets_words %>%
  inner_join(get_sentiments('nrc'),by = 'word')%>%
  select('sentiment')%>%
  group_by(sentiment)%>%
  summarize(freq=n())%>%
  ungroup() %>%
  ggplot(aes(x=reorder(sentiment,desc(freq)),y=freq,fill=freq))+
      geom_bar(position='dodge',stat='identity')+
      xlab('Sentiment')+ylab('Frequency')+coord_flip()
```



Visualizing Text

Word Cloud




```r
library(tidyr); library(wordcloud)
wordcloud_data=
  tweets_words %>%
  anti_join(rbind(stop_words,c('joe','SMART'),c('biden','SMART'),c('joebiden','SMART'),
                  c('https','SMART'),c('t.co','SMART')),by='word')%>%
  count(word,sort=T)%>%
  ungroup()
wordcloud_data= as.data.frame(wordcloud_data)

wordcloud(words = wordcloud_data$word,wordcloud_data$n,scale=c(2,0.5),
        max.words = 150,colors=brewer.pal(9,"Spectral"))
```

```
> ?stop_words
```

Files | Plots | Packages | Help | Viewer

Refresh Help Topic

R: Various lexicons for English stop words | Find in Topic

stop_words {tidytext} | R Documentation

Various lexicons for English stop words

Description

English stop words from three lexicons, as a data frame. The snowball and SMART sets are pulled from the tm package. Note that words with non-ASCII characters have been removed.

Usage

```
stop_words
```

Format

A data frame with 1149 rows and 2 variables:

word

An English word

lexicon

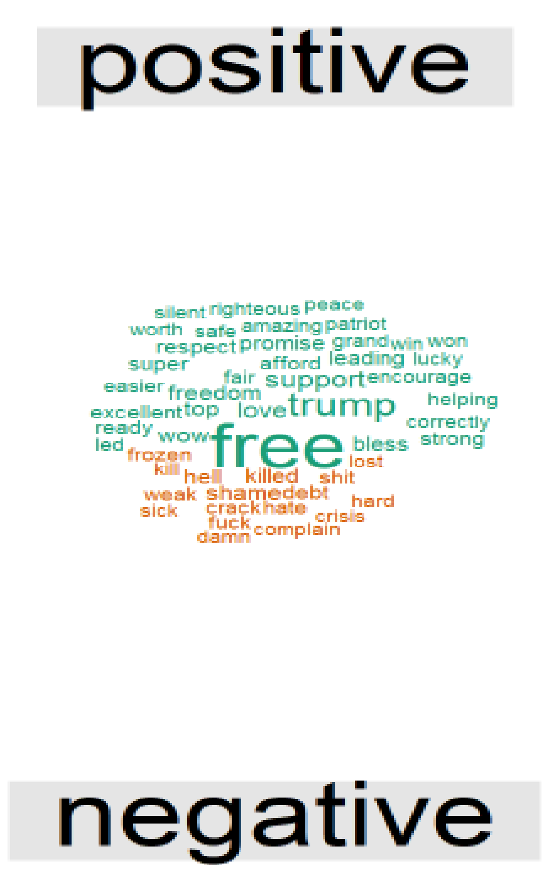
The source of the stop word. Either "onix", "SMART", or "snowball"

```
> stop_words
# A tibble: 1,149 x 2
   word        lexicon
   <chr>       <chr>
 1 a           SMART
 2 a's         SMART
 3 able        SMART
 4 about       SMART
 5 above       SMART
 6 according   SMART
 7 accordingly SMART
 8 across      SMART
 9 actually    SMART
10 after       SMART
```

Comparison Cloud

A comparison cloud comparing Positive words to Negative Words

```
library(tidyr); library(wordcloud)
wordcloud_data=
  tweets_words %>%
  anti_join(stop_words,by='word')%>%
  inner_join(get_sentiments('bing'), by='word')%>%
  count(sentiment,word, sort=T)%>%
  ungroup()%>%
  spread(key=sentiment ,value='n', fill=0)
wordcloud_data= as.data.frame(wordcloud_data)
rownames(wordcloud_data) = wordcloud_data[,'word']
wordcloud_data = wordcloud_data[,c('positive','negative')]
comparison.cloud(wordcloud_data,scale=c(2,0.5), max.words= 50, rot.per=0)
```



Summary

- In this module we,
  - examined the potential of analyzing unstructured data
  - discussed applications of text analysis
  - reviewed various methods used for text analysis
    - examined process of sentiment analysis
    - used text as features in a predictive model
    - worked with twitter data