OPEN DATA SCIENCE CONFERENCE

@ODSC

San Fransisco | November 2017

Ted Kwartler

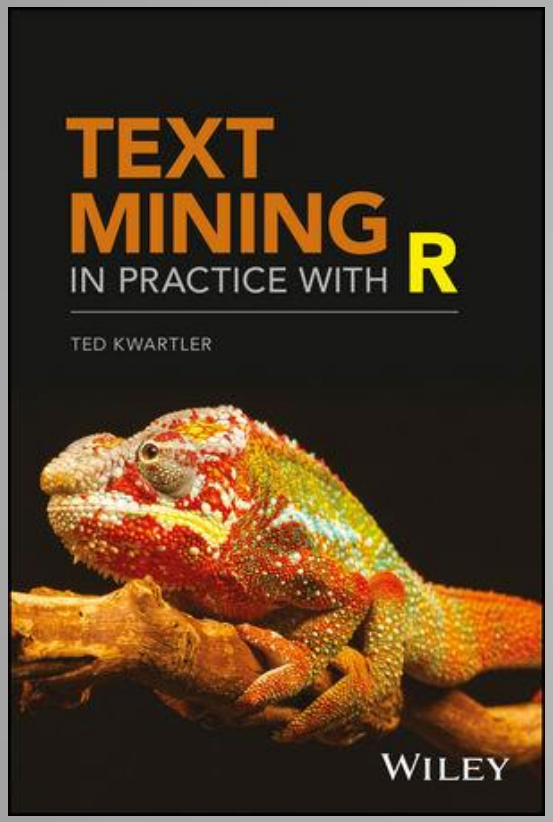
Intro to Text Mining in R

Intro to Natural Language Processing

www.linkedin.com/in/edwardkwartler

@tkwartler

Shameless Plug #1

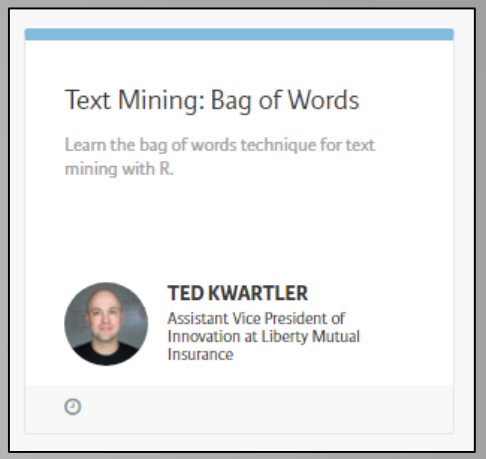


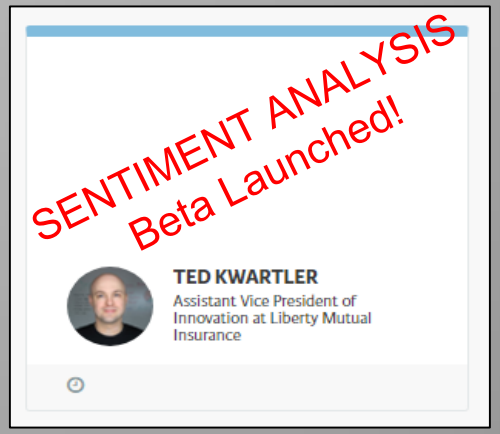
www.TedKwartler.com or
www.amazon.com or
www.wiley.com Discount Code: VBP65

Shameless Plug #2

DataCamp

www.datacamp.com





Workshop Agenda

9-10:45	10:45-11	11 – 11:45	11:45 – 12	12-12:40	12:40-12:50	12:50-1
Intro to TM w/R	Break	Gather Text	Break	HTML Widgets & Flexdashboard	Optional Break	Scheduling

Goals:

- Learn the basics of text mining
- Apply methods to real data
- Follow a business workflow for stakeholders needing text mining insights

Intro to TM w/R Deep Dive

9-10:45	10:45-11	11 – 11:45	11:45 – 12	12-12:40	12:40-12:50	12:50-1
Intro to TM w/R	Break	Gather Text	Break	HTML Widgets & Flexdashboard	Optional Break	Scheduling

- What is Text Mining?
- String Manipulation & Cleaning
- Frequency Counts, Dendrograms & Wordclouds
- Sentiment Analysis
- Topic Modeling
- OpenNLP/Named Entity Recognition

Goals:

- Learn the basics of text mining
- Apply methods to real data and uses
- Follow a business workflow for stakeholders needing text mining insights

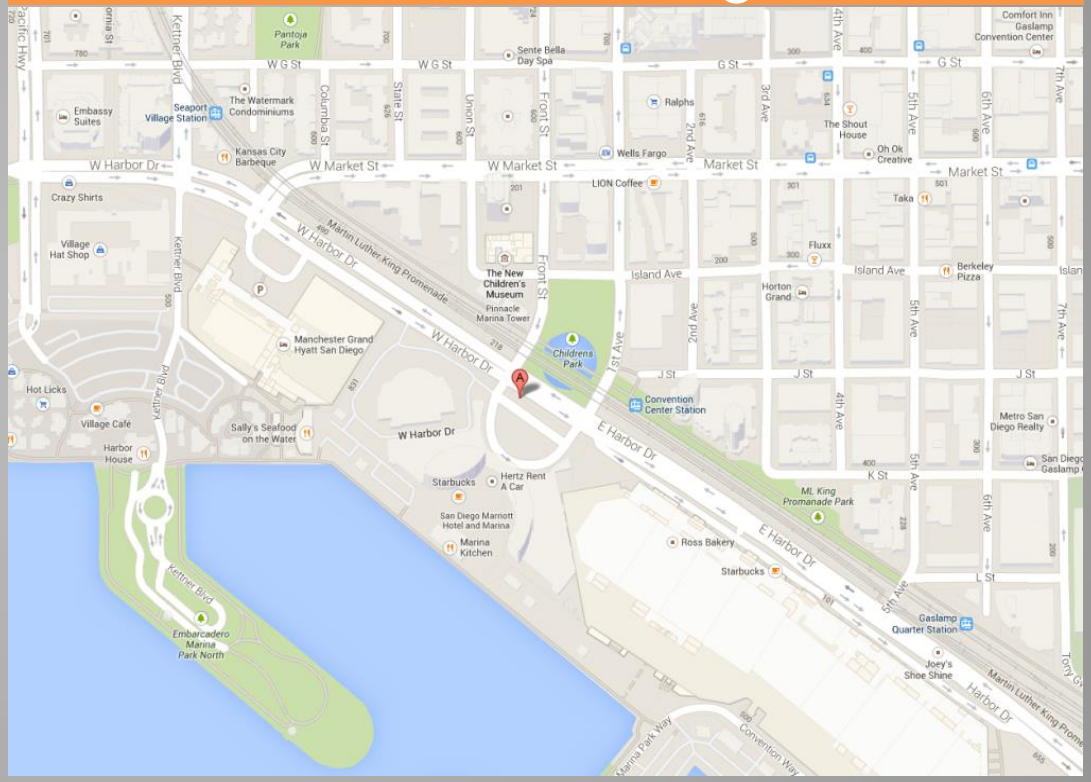
What is Text Mining?

- **Extract new insights from text**
- Let's you drink from a fire hose of information
- Language is hard; many unsolved problems
  - Unstructured
  - Expression is individualistic
  - Multi-language/cultural implications

What is Text Mining?

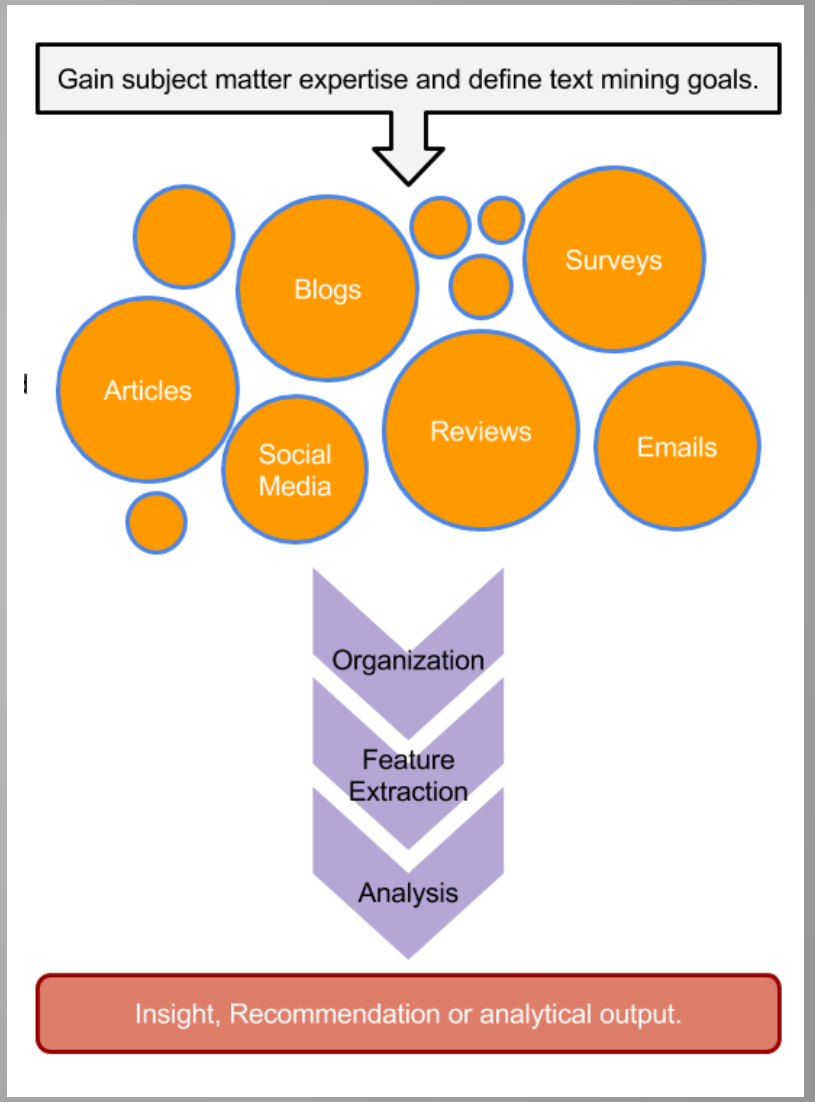
- **Extract new insights from text**
- Let's you drink from a fire hose of information
- Language is hard; many unsolved problems
  - Unstructured
  - Expression is individualistic
  - Multi-language/cultural implications

After Text Mining…



Text Mining Workflow

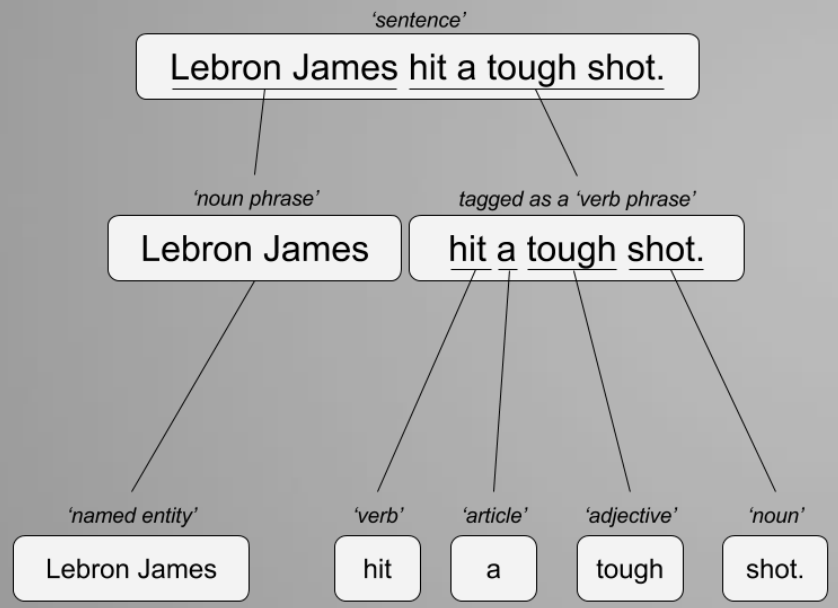
1. Problem Definition
2. Identify Text Sources
3. Text Organization
4. Feature Extraction
5. Analytics
6. Reach Insight or Recommendation



Two Popular Approaches*

"Lebron James hit a tough shot."

Semantic Using Syntactic Parsing



Bag of Words

*There are other approaches usually based on DNN, that I refer to "abstractive"

Text Mining Approaches

Some Challenges in Text Mining

- Compound words (tokenization) changes meaning
  - "not bad" versus "bad"
- Disambiguation
- Sarcasm
  - "I like it…NOT!"
- Cultural differences
  - "It's wicked good" (in Boston)

"I made her duck."

- I cooked waterfowl to eat.
- I cooked waterfowl belonging to her.
- I created the (clay?) duck and gave it to her.
- Duck!!

Text Sources

Text can be captured within the enterprise and elsewhere

- Books
- Electronic Docs (PDFs)
- Blogs
- Websites
- Social Media
- Customer Records
- Customer Service Notes
- Notes
- Emails
- Legal Documents
- …

The source and context of the medium is important. It will have a lot of impact on difficulty and data integrity.

Intro to TM w/R Deep Dive

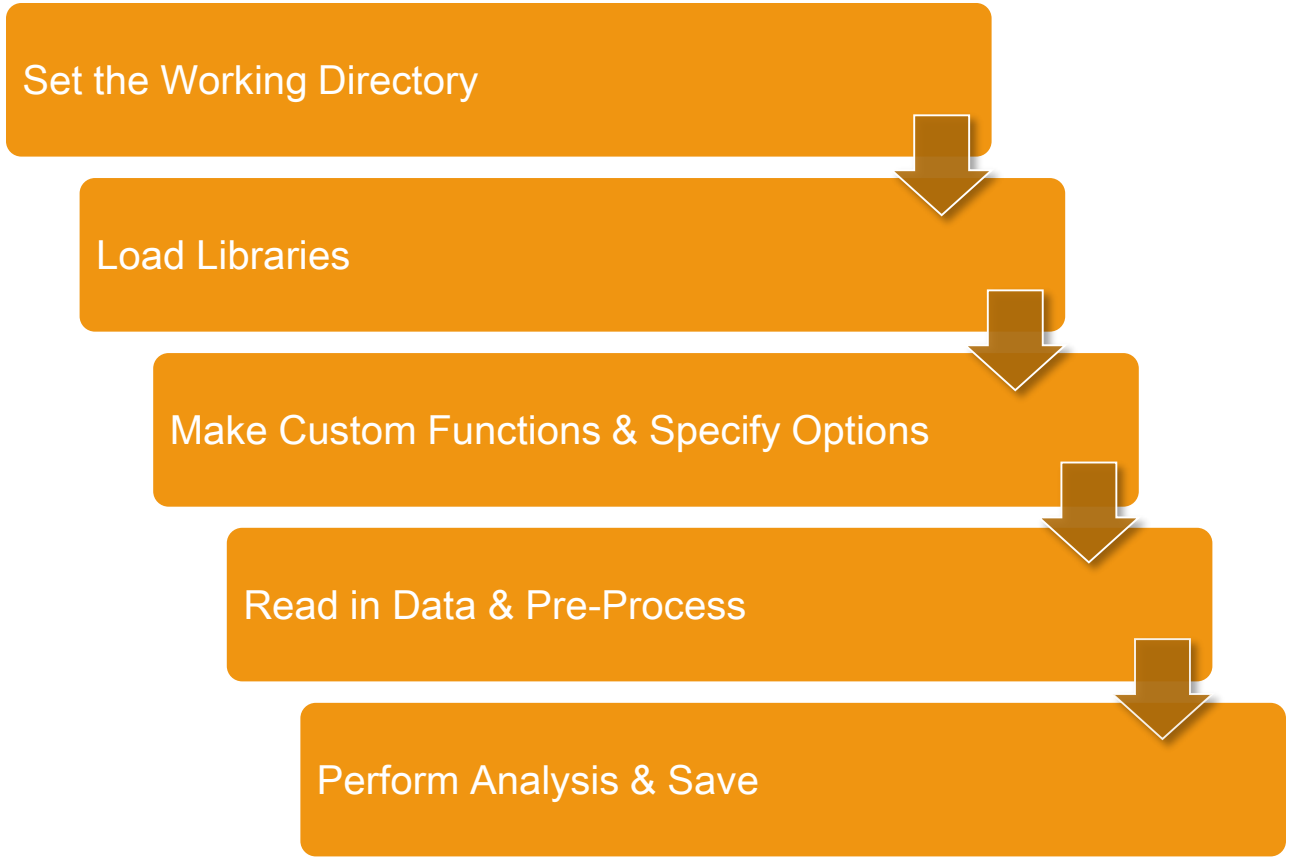
- ~~What is Text Mining?~~
- String Manipulation & Cleaning
- Frequency Counts, Dendrograms & Wordclouds
- Sentiment Analysis
- Topic Modeling
- OpenNLP/Named Entity Recognition

Goals:

- Learn the basics of text mining
- Apply methods to real data and uses
- Follow a business workflow for stakeholders needing text mining insights

Enough of me talking…let's do it for real!

Scripts in this workshop follow a simple workflow



Enough of me talking…let's do it for real!

Setup

```
# Get CRAN packages
libs<-c('stringi', 'tm', 'qdap', 'ggplot2',
        'ggthemes', 'dendextend', 'wordcloud',
        'RColorBrewer', 'tidytext', 'treemap',
        'portfolio','GuardianR','lda','LDAvis',
        'dplyr','servr','SnowballC','openNLP',
        'jsonlite','pbapply','wordcloud2',
        'rbokeh', 'radarchart', 'stringr',
        'flexdashboard', 'rmarkdown', 'httr')

# Get most of the libs
install.packages(libs,
                 repos = "http://cran.r-project.org",
                 dependencies = c("Depends", "Imports", "Suggests"))
```

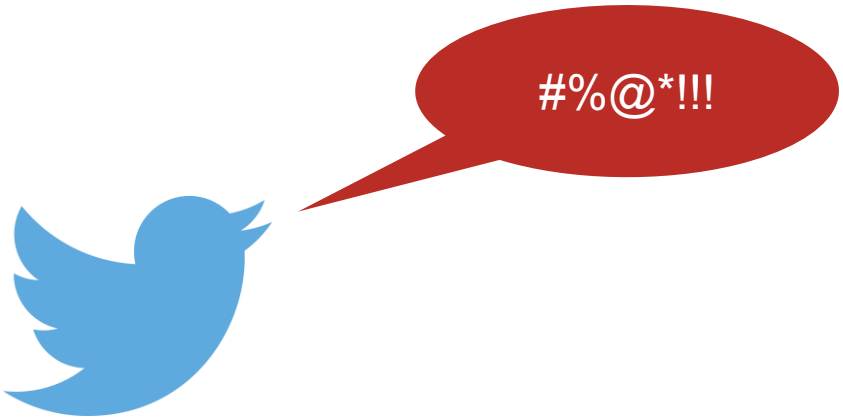
Enough of me talking…let's do it for real!

Setup Continued

```
# WINDOWS Scheduler
install.packages('taskscheduleR')

# LINUX Scheduler
install.packages('cronR')

# Get a NLP Model for use w/openNLP
install.packages('openNLPmodels.en',
                 repos = "http://datacube.wu.ac.at/",
                 type = "source")
```

Warning: Twitter Profanity

- Twitter demographics skew young and as a result have profanity that appear in the examples.
- **It's the easiest place to get a lot of messy text fast, if it is offensive feel free to talk to me and I will work to get you other texts for use on your own. No offense is intended.**



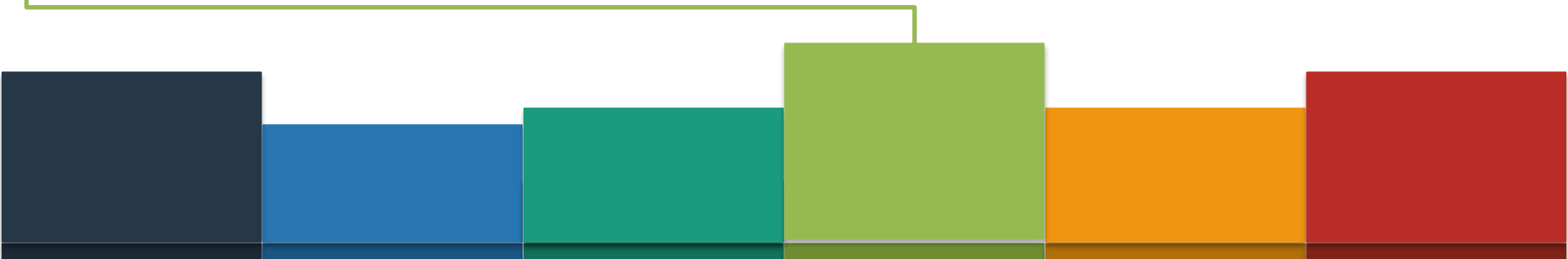
Open the “coffee.csv” to get familiar with the data structure

1000 tweets mentioning “coffee”

	text	favorited	replyToSN	created	truncated	replyToSID	id	replyToUID	statusSource	screenName	retweetCour	retweeted	longitude	latitude
1	@ayyytylerb that is so true drink lots of coffee	FALSE	ayyytylerb	8/9/13 2:43	FALSE	3.6566E+17	3.6566E+17	1637123977	<a href="htt	thejennagibs	0	FALSE	NA	NA
2	RT @bryzy_brib: Senior March tmw morning at 7:25 A.M. in the SENIOR lot. Get up early, make yo coffee/breakfast, cus this will only happen ...	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	<a href="htt	carolynicosia	1	FALSE	NA	NA
3	If you believe in #gunsense tomorrow would be a very good day to have your coffee any place BUT @Starbucks Guns+Coffee=#nosense @MomsDemand	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	web	janeCkay	0	FALSE	NA	NA
4	My cute coffee mug. http://t.co/2udvMU6XIG	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	<a href="htt	AlexandriaO(	0	FALSE	NA	NA
5	RT @slaredo21: I wish we had Starbucks here... Cause coffee dates in the morning sound perff!	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	<a href="htt	Rooosssaaaa	2	FALSE	NA	NA
6	Does anyone ever get a cup of coffee before a cocktail??	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	<a href="htt	E_Z_MAC	0	FALSE	NA	NA
7	"I like my coffee like I like my women...black, bitter, and preferably fair trade." I love #Archer	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	<a href="htt	Charlie_3119	0	FALSE	NA	NA
8	@dreamwwediva ya didn't have coffee did ya?	FALSE	dreamwwed	8/9/13 2:43	FALSE	3.6566E+17	3.6566E+17	1316942208	<a href="htt	JessicaSalvat	0	FALSE	NA	NA
9	RT @iDougherty42: I just want some coffee.	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	<a href="htt	kaytiekirk	1	FALSE	NA	NA
10	RT @Dorkv76: I can't care before coffee.	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	<a href="htt	lissteria	2	FALSE	NA	NA
11	No lie I wouldn't mind coming home smelling like coffee	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	<a href="htt	DOPECROOK	0	FALSE	NA	NA
12	RT @JonasWorldFeed: Play Ping Pong with Joe. Take a tour of the stage with Nick. Have coffee with Kevin. Charity auction: https://t.co/VTkK...	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	<a href="htt	TiffCaruso	6	FALSE	NA	NA
13	Have I ever told any of you that Tate Donovan bought my stepmom coffee?	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	web	CurlysCrazyN	0	FALSE	NA	NA
14	RT @JonasWorldFeed: Play Ping Pong with Joe. Take a tour of the stage with Nick. Have coffee with Kevin. Charity auction: https://t.co/VTkK...	FALSE	NA	8/9/13 2:43	FALSE	NA	3.6566E+17	NA	web	JoeJonasVA	6	FALSE	NA	NA
15	@HeatherWhaley I was about 2 joke it takes 2 hands to hold hot coffee...then I read headline! #Don'tDrinkNShoot	FALSE	HeatherWha	8/9/13 2:42	FALSE	3.6565E+17	3.6566E+17	26035764	<a href="htt	AnnaDuleep	0	FALSE	NA	NA
16	RT @MoveTheSticks: Charlie Whitehurst looks like he should be working at a coffee shop in Portland or hosting a renovation show on HGTV.	FALSE	NA	8/9/13 2:42	FALSE	NA	3.6566E+17	NA	<a href="htt	mpr4437	42	FALSE	NA	NA
17	Coffee always makes everything better.	FALSE	NA	8/9/13 2:42	FALSE	NA	3.6566E+17	NA	web	sharkshukri	0	FALSE	NA	NA
18	RT @AdelaideReview: Food For Thought: @Annabelleats shares a delicious Venison and Porcini Mushroom Pie Recipe. http://t.co/N8O7vqFKWN http...	FALSE	NA	8/9/13 2:42	FALSE	NA	3.6566E+17	NA	<a href="htt	thepaulbake	1	FALSE	NA	NA
19	RT @LittleMelss: lmfao!!!"@bryanlaca: nahhh Melanie u is fa sho like an ummm a Coffee table ;)) yeeeee lmaoo"	FALSE	NA	8/9/13 2:42	FALSE	NA	3.6566E+17	NA	web	bryanlaca	1	FALSE	NA	NA
20	I wonder if Christian Colon will get a cup of coffee once the rosters expand to 40 man in September. Really nothing to lose by doing so.	FALSE	NA	8/9/13 2:42	FALSE	NA	3.6566E+17	NA	<a href="htt	Shauncore	0	FALSE	NA	NA

“text$text”
is the vector of tweets that we are interested in.

All other attributes are automatically returned from the twitter API



1_Keyword_Scanning.R

Basic R Unix Commands

grepl returns a vector of T/F if the pattern is present at least once

```
grepl("pattern", searchable object, ignore.case=TRUE)
```

```
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[18] FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

grep returns the position of the pattern in the document

```
grep("pattern", searchable object, ignore.case=TRUE)
```

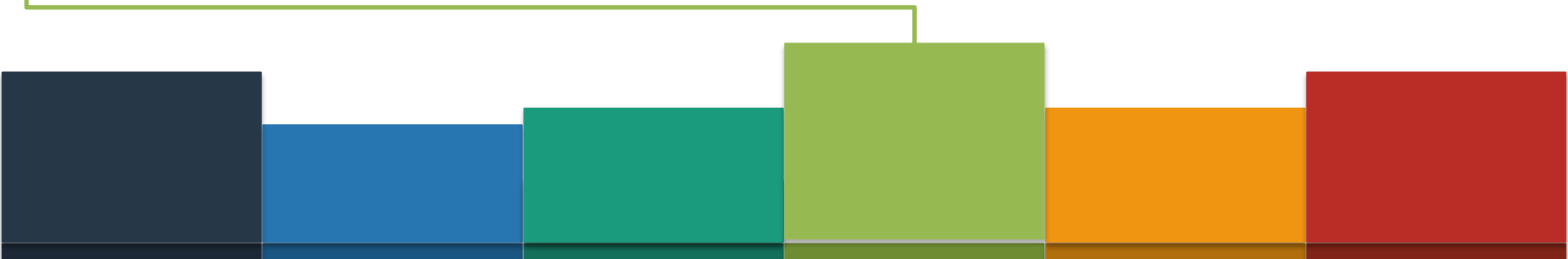
```
[1]   4 214 276 366 479 534 549 620
```

"library(stringi)" Functions

stri_count counts the number of patterns in a document

```
stri_count(searchable object, fixed="pattern")
```

```
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[54] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
```



1_Keyword_Scanning

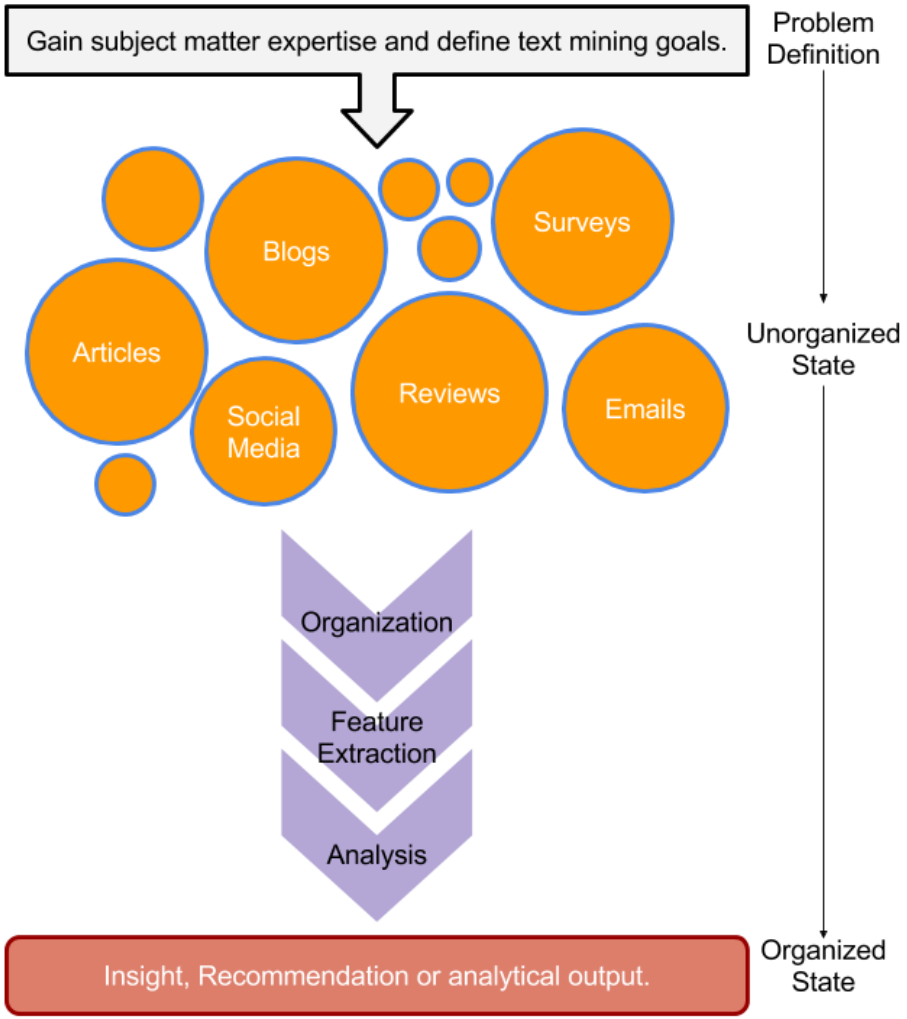
Intro to TM w/R Deep Dive

- ~~What is Text Mining?~~
- ~~String Manipulation~~ & Cleaning
- Frequency Counts, Dendrograms & Wordclouds
- Sentiment Analysis
- Topic Modeling
- OpenNLP/Named Entity Recognition

Goals:

- Learn the basics of text mining
- Apply methods to real data and uses
- Follow a business workflow for stakeholders needing text mining insights

Remember This?

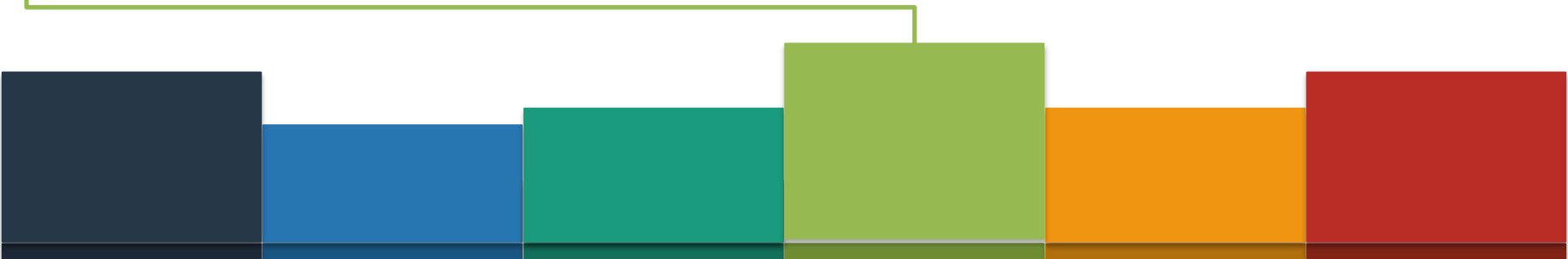


R for our Cleaning Steps

Tomorrow I'm going to have a nice glass of Chardonnay and wind down with a good book in the corner of the county :-)

1. Remove Punctuation
2. Remove extra white space
3. Remove Numbers
4. Make Lower Case
5. Remove "stop" words

tomorrow going nice glass chardonnay wind down good book corner county



2_Cleaning_and_Frequency_Count.R

"library(tm)" Functions

VCorpus creates a corpus held in memory.

```
VCorpus(source)
```

tm_map applies the transformations for the cleaning

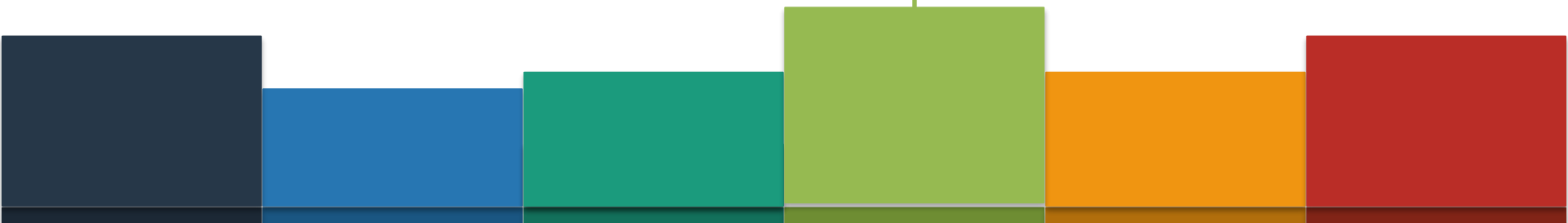
```
tm_map(corpus, function)
```

getTransformations() will list all standard tm corpus transformations
We can apply standard R ones too. Sometimes it makes sense to perform all of these or a subset or even other transformations not listed like "stemming"

```
tm_map(corpus, removePunctuation) - removes the punctuation from the documents
      tm_map(corpus, stripWhitespace) - extra spaces, tabs are removed
            tm_map(corpus, removeNumbers) - removes numbers
     tm_map(corpus, content_transformer(tolower)) – makes all case lower
         tm_map(corpus, removeWords) - removes specific "stopwords"
```

New Text Mining Concepts

Corpus- A collection of documents that analysis will be based on.
Stopwords – are common words that provide very little insight, often articles like "a", "the". Customizing them is sometimes key in order to extract valuable insights.



2_Cleaning_and_Frequency_Count.R

"library(qdap)" Functions

Multiple Global Substitution

```
mgsub("search pattern", "replacement pattern", text object)
```

Family of Replace Functions

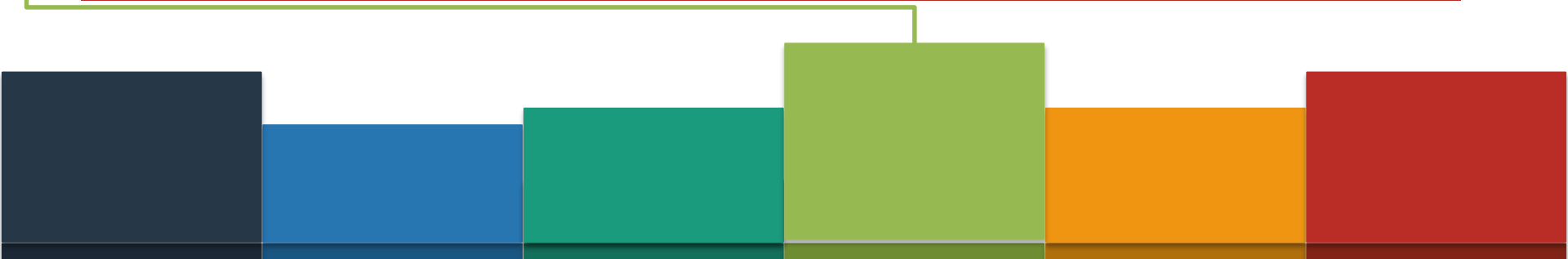
```
replace_abbreviation()-  Replace Abbreviations
replace_contraction()- Replace Contractions
replace_number()-  Replace Numbers With Text Representation
replace_ordinal()- Replace Mixed Ordinal Numbers With Text Representation
replace_symbol()- Replace Symbols with Word Equivalents
```

To use on a corpus you need to apply content_transformer

```
tm_map(corpus, content_transformer(replace_abbreviation))
```

New Text Mining Concepts

Lemmatization in linguistics, is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item.



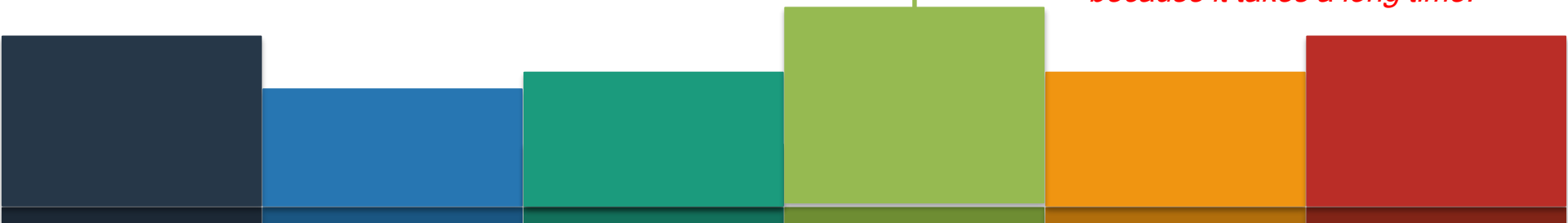
Poor Man's lemmatization

```
library(lexicon)
> hash_lemmas
             token    lemma
     1:  furtherst  further
     2:    skilled    skill
     3:     'cause  because
     4:         'd    would
     5:        'em     them
    ---
 41529:       zoos      zoo
 41530:  zoospores zoospore
 41531:  zucchinis zucchini
 41532:      zulus     zulu
 41533:    zygotes   zygote
```

Qdap's mgsub() lets you easily aggregate words

```
#Poor Man's Lemmatization
data(hash_lemmas)
text$text <- mgsub(hash_lemmas$token,hash_lemmas$lemma,text$text)
```

Warning,: Not done in the workshop because it takes a long time.



2_Cleaning_and_Frequency_Count.R

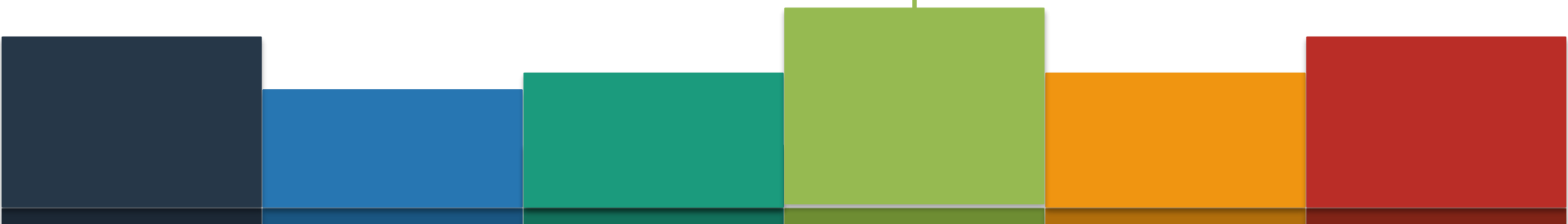
"tryTolower"is poached to account for errors when making lowercase.

```
tryTolower <- function(x){
  # return NA when there is an error
  y = NA
  # tryCatch error
  try_error = tryCatch(tolower(x), error =
function(e) e)
  # if not an error
  if (!inherits(try_error, 'error'))
    y = tolower(x)
  return(y)
}
```

"clean.corpus" makes applying all transformations easier.

```
cleanCorpus<-function(corpus){
  corpus <- tm_map(corpus,
content_transformer(qdapRegex::rm_url))
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, stripWhitespace)
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus,
content_transformer(tryTolower))
  corpus <- tm_map(corpus, removeWords, customStopwords)
  return(corpus)
}
```

Base: tolower (basic)
Stringr: str_to_lower (wrapper)
Custom: tryTolower (handles errors)



2_Cleaning_and_Frequency_Count.R

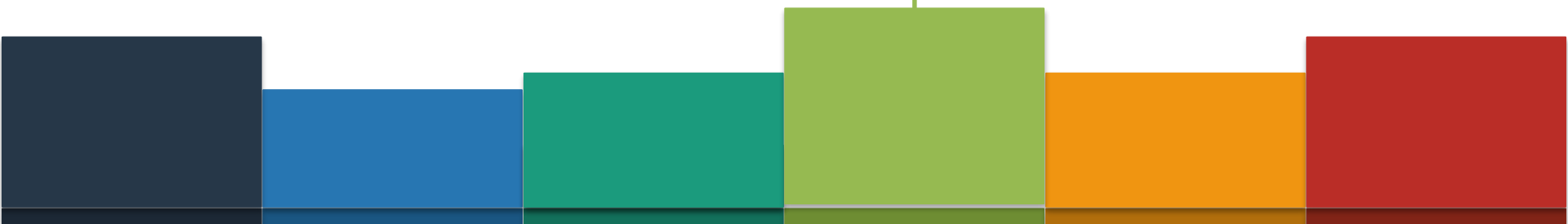
"custom.stopwords" combines vectors of words to remove from the corpus

```
#Create custom stop words
custom.stopwords <- c(stopwords('english'), 'lol', 'smh')
```

Add channel specific stop words.
E.g. Twitter abbreviations

"custom.reader" keeps the meta data (tweet ID) with the original document

```
# Data
text<-read.csv('coffee.csv', header=TRUE)

# Keep the meta data, apply the functions to make a clean corpus
customReader <- readTabular(mapping=list(content="text",
id="id"))
txtCorpus <- VCorpus(DataframeSource(text),
readerControl=list(reader=customReader))
txtCorpus<-cleanCorpus(txtCorpus)
```



2_Cleaning_and_Frequency_Count.R

Bag of Words means creating a Term Document Matrix or Document Term Matrix*

Term Document Matrix

	Tweet1	Tweet 2	Tweet3	Tweet4	…	Tweet_n
Term1	0	0	0	0	0	0
Term2	1	1	0	0	0	0
Term3	1	0	0	2	0	0
…	0	0	3	0	1	1
Term_n	0	0	0	1	1	0

Document Term Matrix

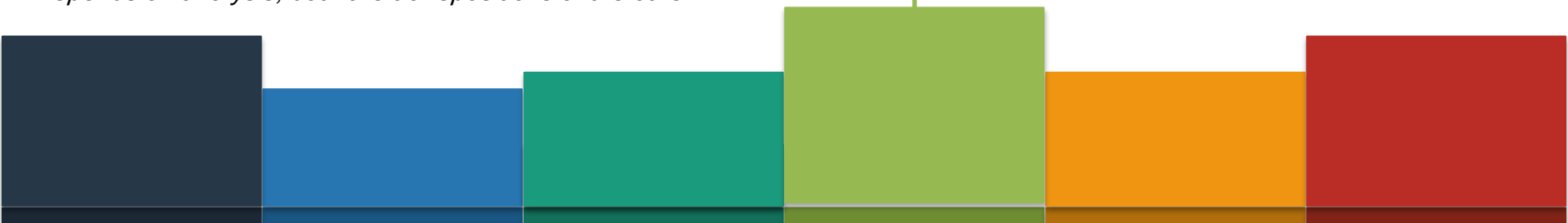
	Term1	Term2	Term3	…	Term_n
Tweet1	0	1	1	0	0
Tweet2	0	1	0	0	0
Tweet3	0	0	0	3	0
…	0	0	0	1	1
Tweet_n	0	0	0	1	0

"as.matrix" makes the tm's version of a matrix into a simpler version

```
txtDtm<-DocumentTermMatrix(txtCorpus)
txtTdm<-TermDocumentMatrix(txtCorpus)
txtDtmM<-as.matrix(txtDtm)
txtTdmM<-as.matrix(txtTdm)
```

These matrices are often very sparse and large therefore some special steps may be needed and will be covered in subsequent scripts.

**Depends on analysis, both are transpositions of the other*



2_Cleaning_and_Frequency_Count.R

Intro to TM w/R Deep Dive

- ~~What is Text Mining?~~
- ~~String Manipulation & Cleaning~~
- Frequency Counts, Dendrograms & Wordclouds
- Sentiment Analysis
- Topic Modeling
- OpenNLP/Named Entity Recognition

Goals:

- Learn the basics of text mining
- Apply methods to real data and uses
- Follow a business workflow for stakeholders needing text mining insights

3_Dendrogram.R

First let's explore simple frequencies

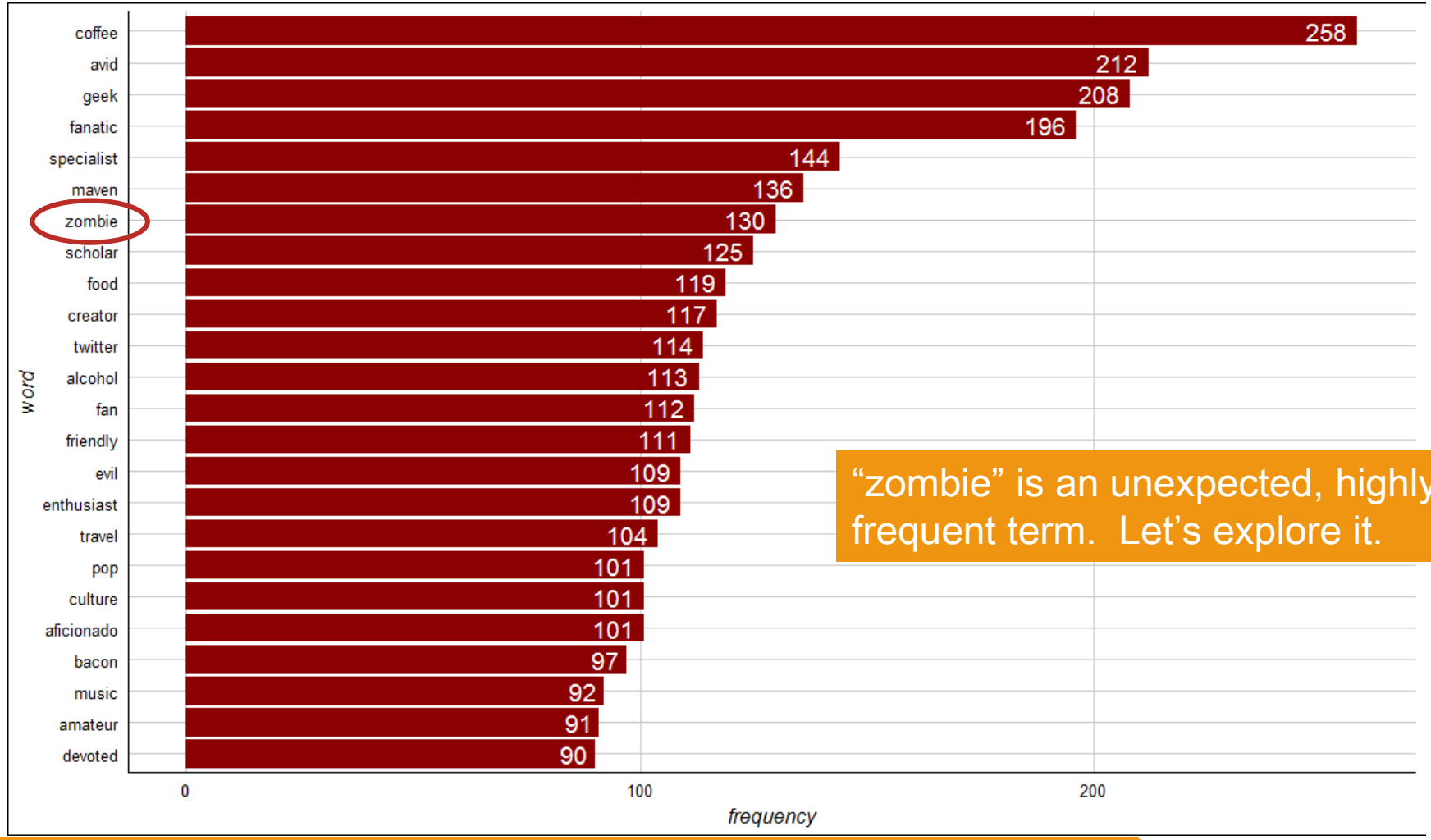
```
beerFreq<-rowSums(beerTDMm)
beerFreq<-data.frame(word=names(beerFreq),frequency=beerFreq)
```

Term Document Matrix

	Tweet1	Tweet 2	Tweet3	Tweet4	…	Tweet_n
Term1	0	0	0	0	0	0
Term2	1	1	0	0	0	0
Term3	1	0	0	2	0	0
…	0	0	3	0	1	1
Term_n	0	0	0	1	1	0

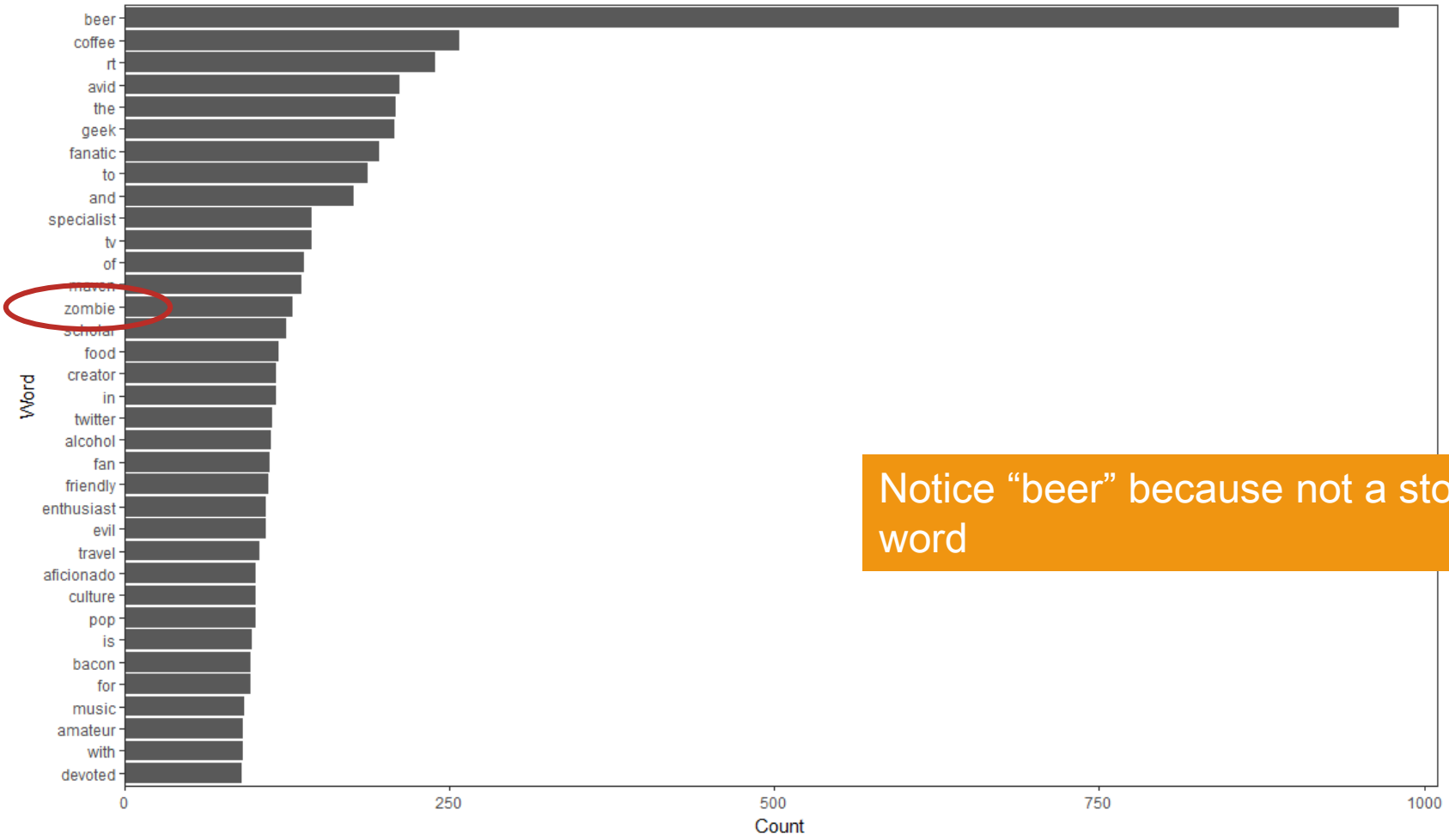
word	freq
Term1	0
Term2	2
Term3	3
…	5
Term_n	2

3_Dendrogram.R - ggplot



"zombie" is an unexpected, highly frequent term. Let's explore it.

3_Dendrogram.R - qdap



Notice “beer” because not a stop word

Freq Counts, Dendrograms Wordclouds

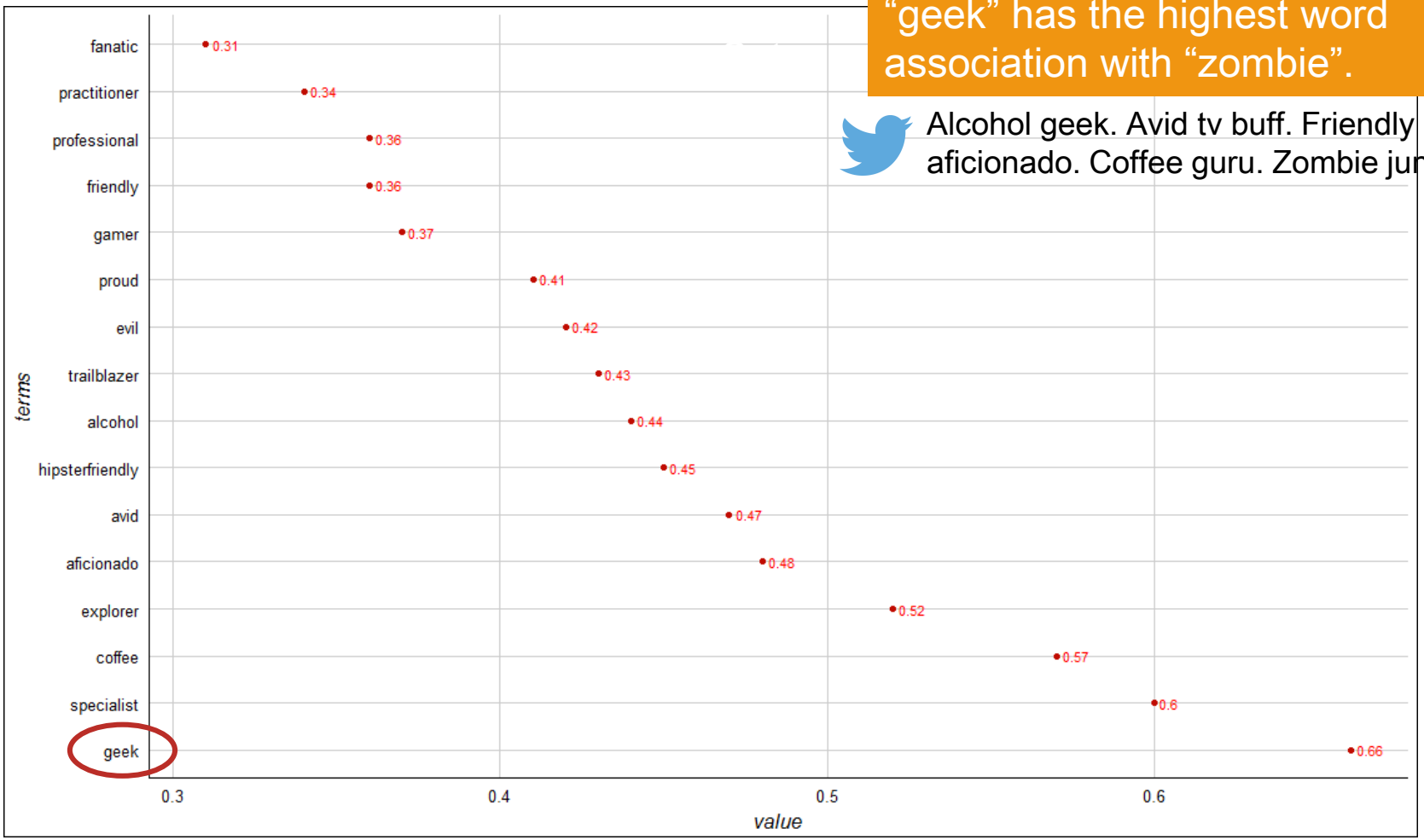
3_Dendrogram.R

Next let's explore word associations, similar to correlation

```
# Inspect word associations
associations<-findAssocs(beerTDM, 'zombie', 0.30)
str(associations)

# Make a dot plot
zombieDF<-data.frame(terms=names(associations[[1]]),
                     value=unlist(associations))
zombieDF$terms<-factor(zombieDF$terms, levels=zombieDF$terms)

ggplot(zombieDF, aes(y=terms)) +
  geom_point(aes(x=value), data=zombieDF, col='#c00c00') +
  theme_gdocs() +
  geom_text(aes(x=value,label=value), colour="red",hjust=-.25,
size=3)
```

- Adjust 0.30 to get the terms that are associated .30 or more with the 'zombie' term.
- Treating the terms as factors lets ggplot2 sort them for a cleaner look.

3_Dendrogram.R



"geek" has the highest word association with "zombie".

Alcohol geek. Avid tv buff. Friendly beer aficionado. Coffee guru. Zombie junkie.

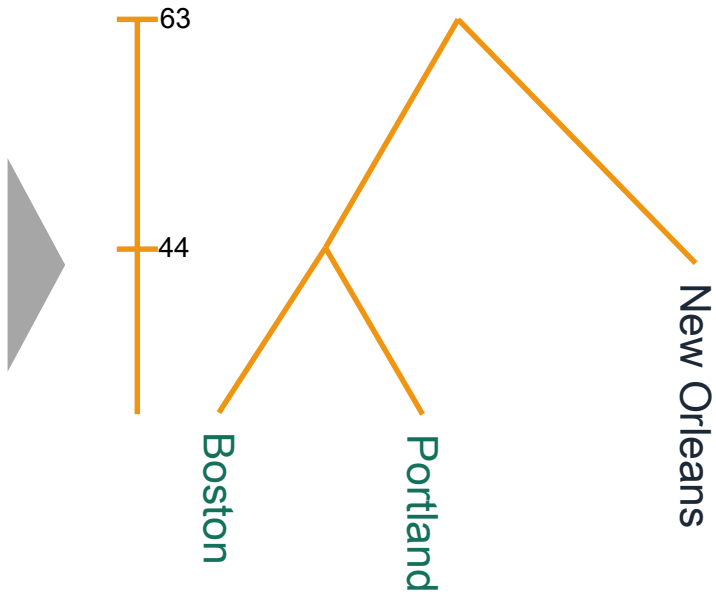
Extracting Meaning using dendrograms

Dendrograms visualize hierarchical clusters based on frequencies distances.

- Reduces information much like average is a reduction of many observations' values
- Word clusters emerge often showing related terms
- Term frequency is used to construct the word cluster. Put another way, term A and term B have similar frequencies in the matrix so they are considered a cluster.

City	Annual Rainfall
Portland	43.5
Boston	43.8
New Orleans	62.7

Boston & Portland are a cluster at height 44. You lose some of the exact rainfall amount in order to cluster them.



3_Dendrogram.R

Weird associations! Maybe a dendrogram will help us more

```
# Reduce TDM
beerTDM2 <- removeSparseTerms(beerTDM, sparse=0.97) #shoot for ~50 terms
beerTDM2<-as.data.frame(as.matrix(beerTDM2))
```

% of zeros allowed
e.g. higher means more words in TDM/DTM

```
> beerTDM
<<TermDocumentMatrix (terms: 2352, documents: 1000)>>
Non-/sparse entries: 10132/2341868
Sparsity           : 100%
Maximal term length: 65
Weighting          : term frequency (tf)
```

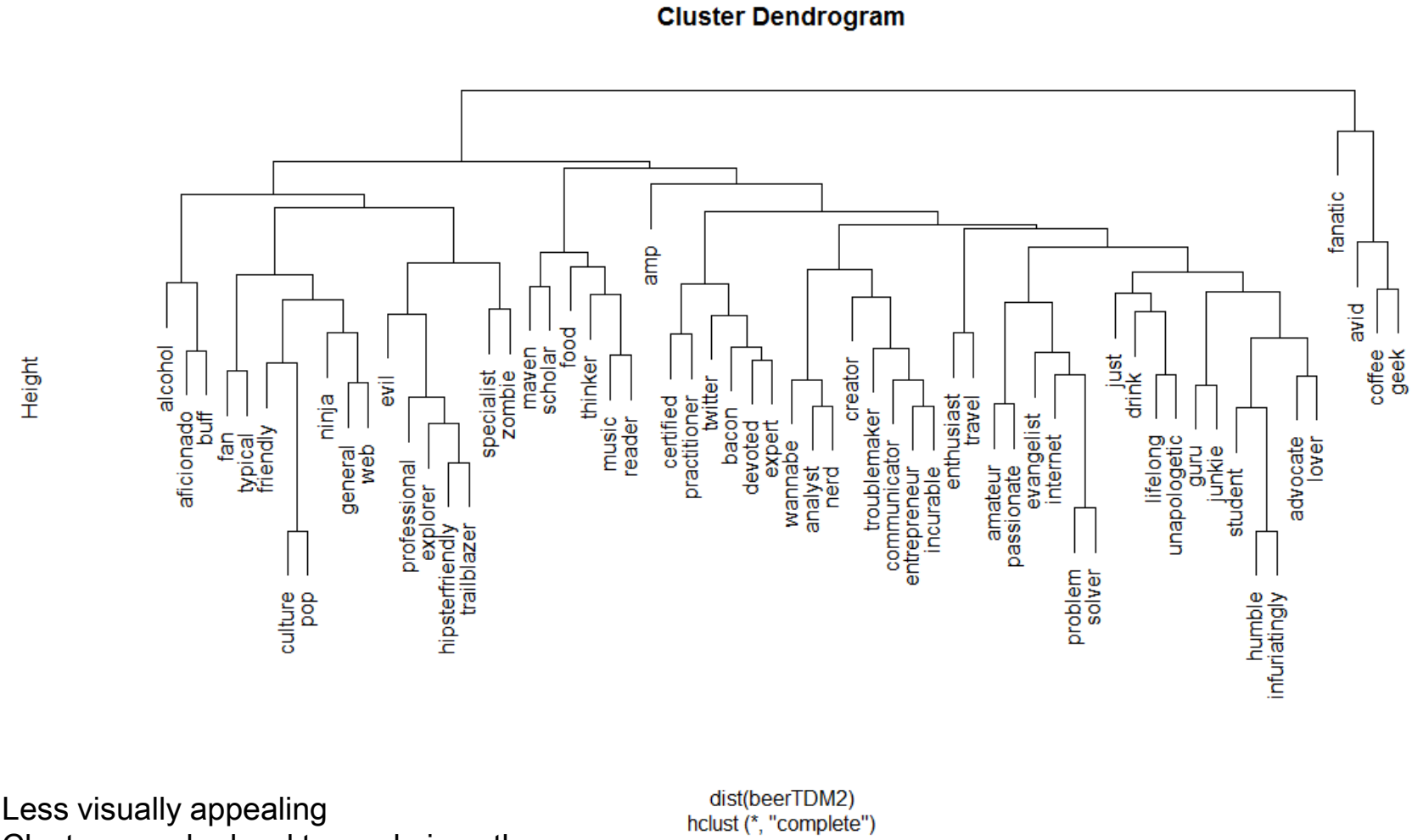
```
> beerTDM2
<<TermDocumentMatrix (terms: 62, documents: 1000)>>
Non-/sparse entries: 5242/56758
Sparsity           : 92%
Maximal term length: 15
Weighting          : term frequency (tf)
```

New Text Mining Concept

Sparse- Term Document Matrices are often extremely sparse. This means that any document (column) has mostly zero's. Reducing the dimensions of these matrices is possible by specifying a sparse cutoff parameter. Higher sparse parameter will bring in more terms.

3_Dendrogram.R

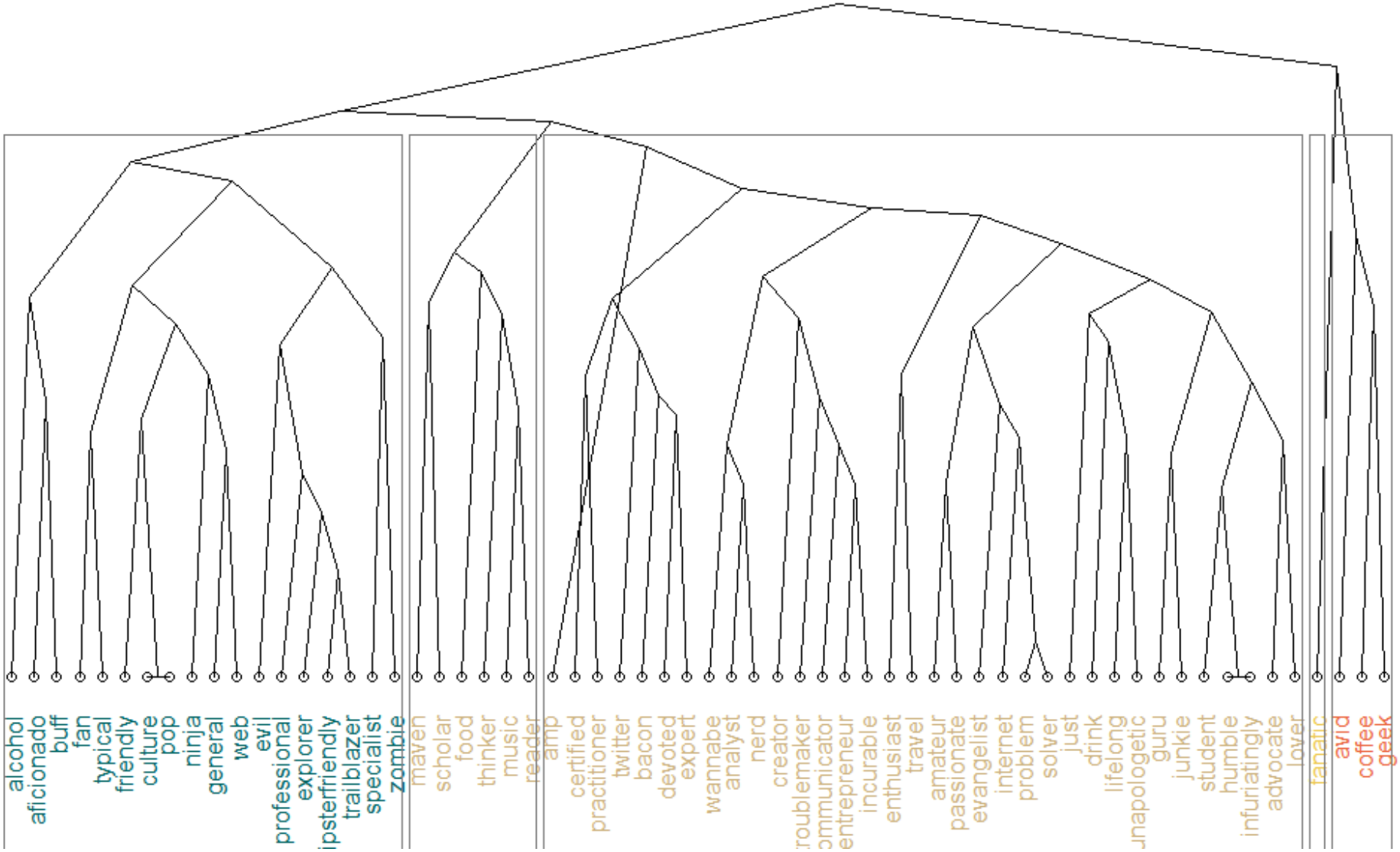
Base Plot of a Dendrogram



- Less visually appealing
- Clusters can be hard to read given the different heights

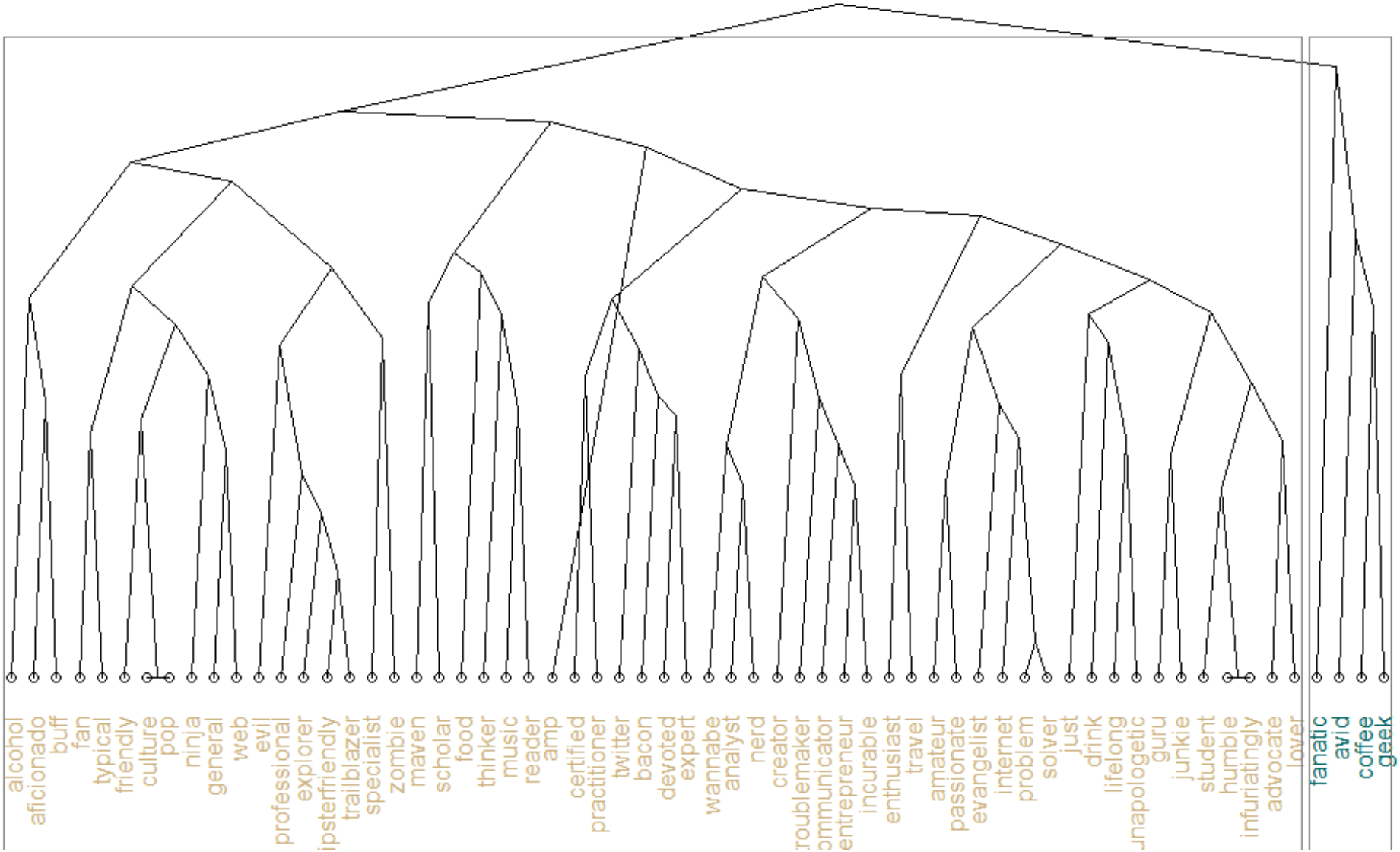
3_Dendrogram.R

Dendextend offers more flexibility



3_Dendrogram.R

Dendextend offers more flexibility



3_Dendrogram.R

Intro to TM w/R Deep Dive

- ~~What is Text Mining?~~
- ~~String Manipulation & Cleaning~~
- ~~Frequency Counts, Dendrograms~~ & Wordclouds
- Sentiment Analysis
- Topic Modeling
- OpenNLP/Named Entity Recognition

Goals:

- Learn the basics of text mining
- Apply methods to real data and uses
- Follow a business workflow for stakeholders needing text mining insights

4_Simple_Wordcloud.R

```
#bigram token maker
bigramTokens <-function(x)
  unlist(lapply(NLP::ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)
```

```
wineTDM<-TermDocumentMatrix(txtCorpus, control=list(tokenize=bigramTokens))
```

Text Mining is so fun. So do Text Mining!

Unigram

Terms	Docs
fun.	1
mining	2
text	2

Bigram

Terms	Docs
do text	1
fun so	1
is so	1
mining is	1
so do	1
so fun	1
text mining	2

**with common stopwords*

New Text Mining Concept

Tokenization- So far we have created single word n-grams. We can create multi word "tokens" like bigrams, or trigrams with this line function. It is applied when making the term document matrix.

4_Simple_Wordcloud.R

To make a wordcloud we follow the previous steps and create a data frame with the word and the frequency.

```
# Get Row Sums
wineTDMv <- sort(rowSums(wineTDMm),decreasing=TRUE)
wineDF <- data.frame(word = names(wineTDMv),freq=wineTDMv)
```

Term Document Matrix

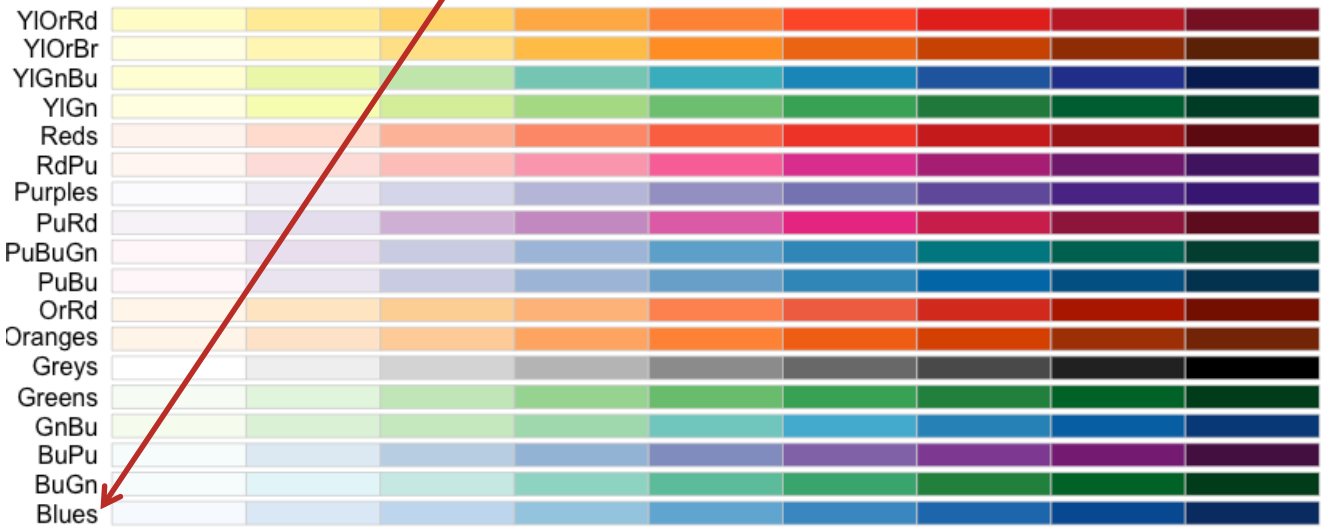
	Tweet1	Tweet 2	Tweet3	Tweet4	…	Tweet_n
Term1	0	0	0	0	0	0
Term2	1	1	0	0	0	0
Term3	1	0	0	2	0	0
…	0	0	3	0	1	1
Term_n	0	0	0	1	1	0

word	freq
Term1	0
Term2	2
Term3	3
…	5
Term_n	2

4_Simple_Wordcloud.R

Next we need to select the colors for the wordcloud.

```
# Review all Pallettes
display.brewer.all()

# Choose a color & drop light ones
pal <- brewer.pal(8, "Blues")
pal <- pal[-(1:2)]
```



4_Simple_Wordcloud.R

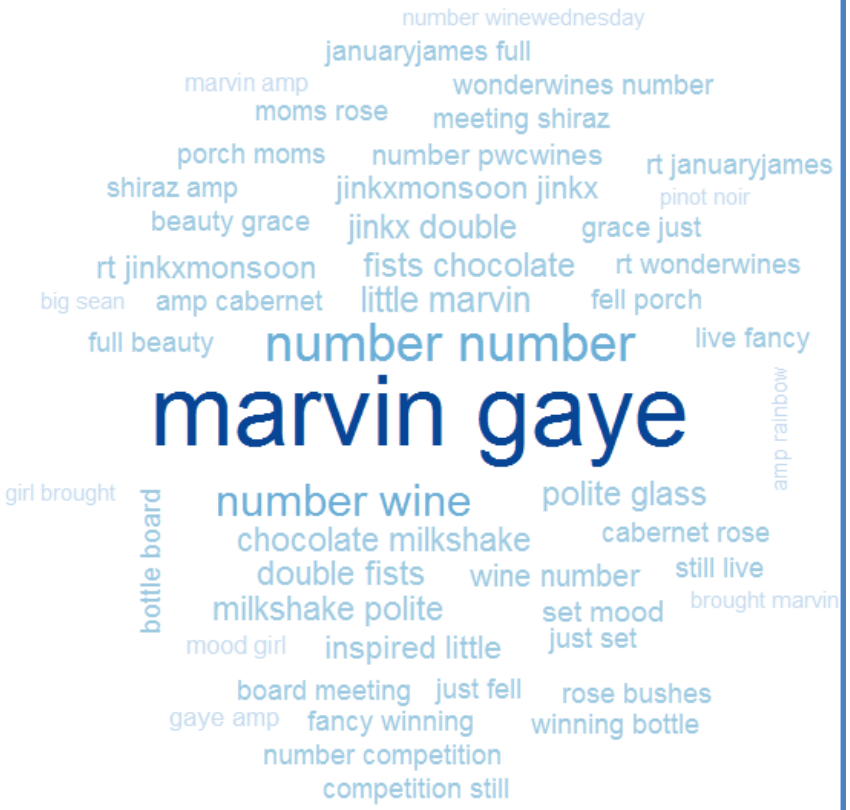
4_Simple_Wordcloud.R

```
set.seed(1234)
wordcloud(wineDF$word,wineDF$freq,max.words=50, random.order=FALSE, colors=pal)
```

- Bigram Tokenization has captured “marvin gaye”
- A word cloud is a frequency visualization. The larger the term (or bigram here) the more frequent the term.
- You may get warnings if certain tokens are to large to be plotted in the graphics device.
- In cleanCopurs() the function … changes numeric with the generic string “number” so be careful with your preprocessing steps!

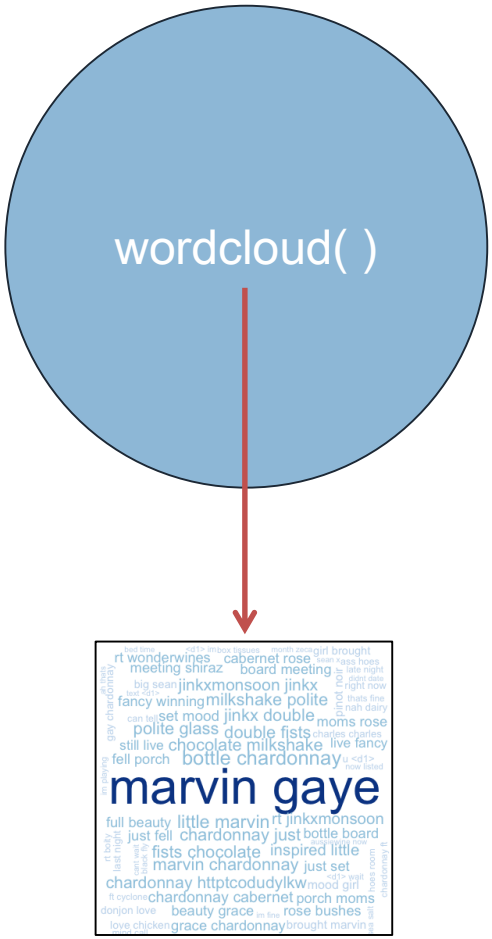
4_Simple_Wordcloud.R

In cleanCopurs() the function replace_symbol() changes numeric with the generic string “number” so be careful with your preprocessing steps!

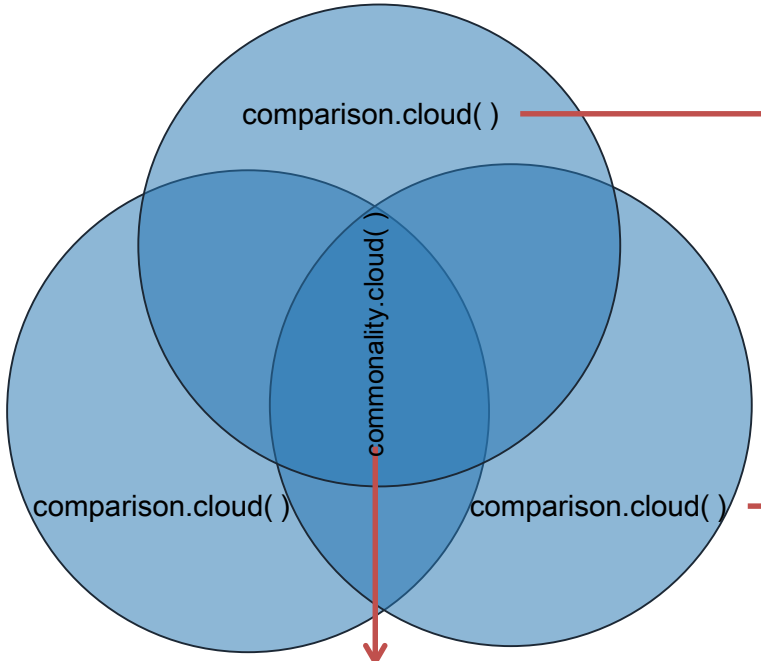


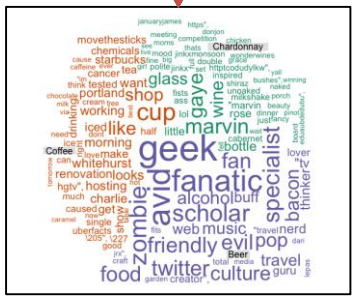
Types of Wordclouds

Single Corpus



Multiple Corpora





5_Other_Wordclouds.R

The challenge is working with multiple corpora efficiently. Many ways to do it…

```
# Read in multiple files as individuals
txtFiles<-c('chardonnay.csv','coffee.csv','beer.csv') #use list.files() for a
lot
for (i in 1:length(txtFiles)){
  assign(txtFiles[i], read.csv(txtFiles[i]))
  cat(paste('read completed:',txtFiles[i],'\n'))
}

# Read them into a single list with individual elements
all<-pblapply(txtFiles,read.csv)
```

Two ways to import csv files:

1. Each file is read in and an object created for each.
2. A list called “all” is created. Each list element represents a single document. Using data.table::rbindlist() one can create a single document from all files in the folder.

5_Other_Wordclouds.R

Commonality Cloud

- The tweets mentioning “chardonnay” “beer”, and “coffee” have these words in common.
- Again size is related to frequency.
- Not helpful in this but in diverse corpora it may be more helpful e.g. political speeches.


```
# Make commonality cloud
commonality.cloud(drinkTDMm, max.words=150, random.order=FALSE,colors=pal)
```

Comparison Cloud

- The tweets mentioning “chardonnay” “beer”, and “coffee” have these dissimilar words.
- Again size is related to frequency.
- Beer drinkers in this snapshot are passionate (fanatics, geeks, specialists) on various subjects while Chardonnay drinkers mention Marvin Gaye. Coffee mentions up & working.


```
# Make comparison cloud
comparison.cloud(drinkTDMm, max.words=75, random.order=FALSE,
title.size=0.5,colors=brewer.pal(ncol(drinkTDMm),"Dark2"))
```

Intro to TM w/R Deep Dive

- ~~What is Text Mining?~~
- ~~String Manipulation & Cleaning~~
- ~~Frequency Counts, Dendrograms & Wordclouds~~
- Sentiment Analysis
- Topic Modeling
- OpenNLP/Named Entity Recognition

Goals:

- Learn the basics of text mining
- Apply methods to real data and uses
- Follow a business workflow for stakeholders needing text mining insights

Simple Sentiment Polarity

Scoring

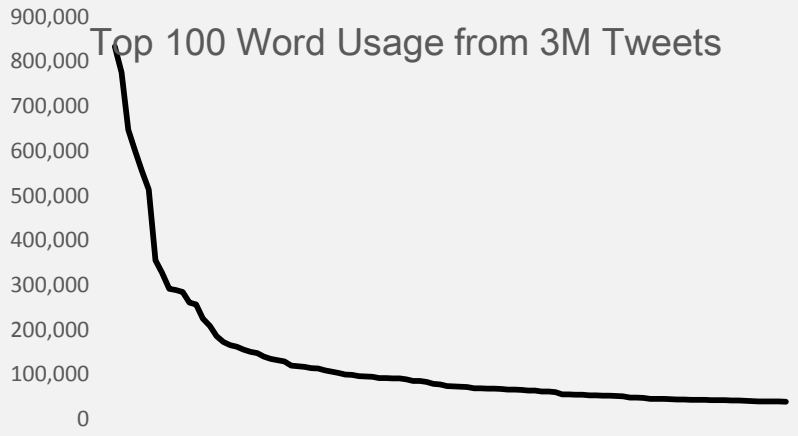
Surprise is a sentiment.
Hit by a bus! – Negative Polarity
Won the lottery!- Positive Polarity

- I loathe BestBuy Service -1
- I love BestBuy Service. They are the best. +2
- I like shopping at BestBuy but hate traffic. 0

R's QDAP polarity function scans for positive words, and negative words as defined by MQPA Academic Lexicon research. It adds positive words and subtracts negative ones along with valence shifters. The final score represents the polarity of the social interaction.

Zipf's Law

Many words in natural language but there is steep decline in everyday usage. Follows a predictable pattern.



Simple Sentiment Polarity

Scoring

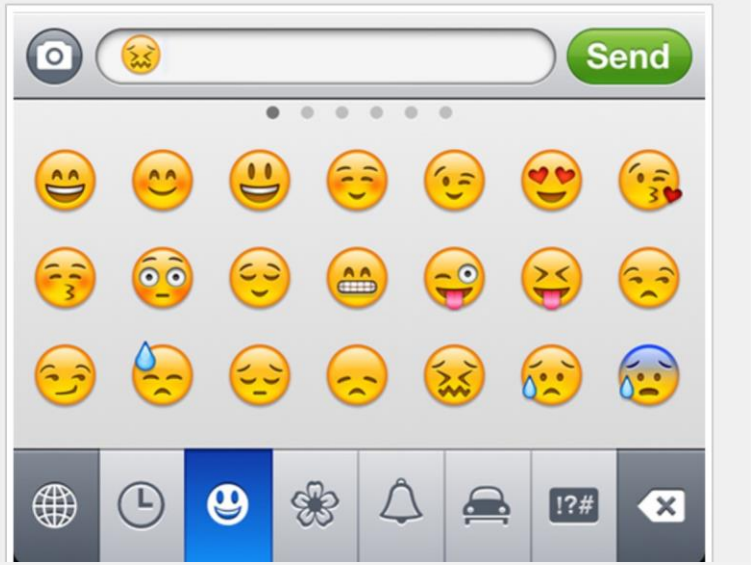
```
library(qdap)

text1<-'i love St Peters University'
text2<-'this lecture is good'
text3<-'this lecture is very good'
text4<-'data science is hard I like it a little'
text5<-'data science is hard'

polarity(text1)
polarity(text2)
polarity(text3)
polarity(text4)
polarity(text5)
```

- Text 1: "love" was identified as positive. The text has 5 words and so 1/sqrt(5) = .447
- Text 2: "good" was identified positively. So 1/sqrt(4)=.5
- Text 3: "good" was found along with the amplifier "very". So (.8+1)/sqrt(5)=.805
- Text 4: hard and like cancel each other out so the polarity is zero. 1-1/sqrt(9)=0
- Text 5: "hard" is -1/sqrt(4)=-.50

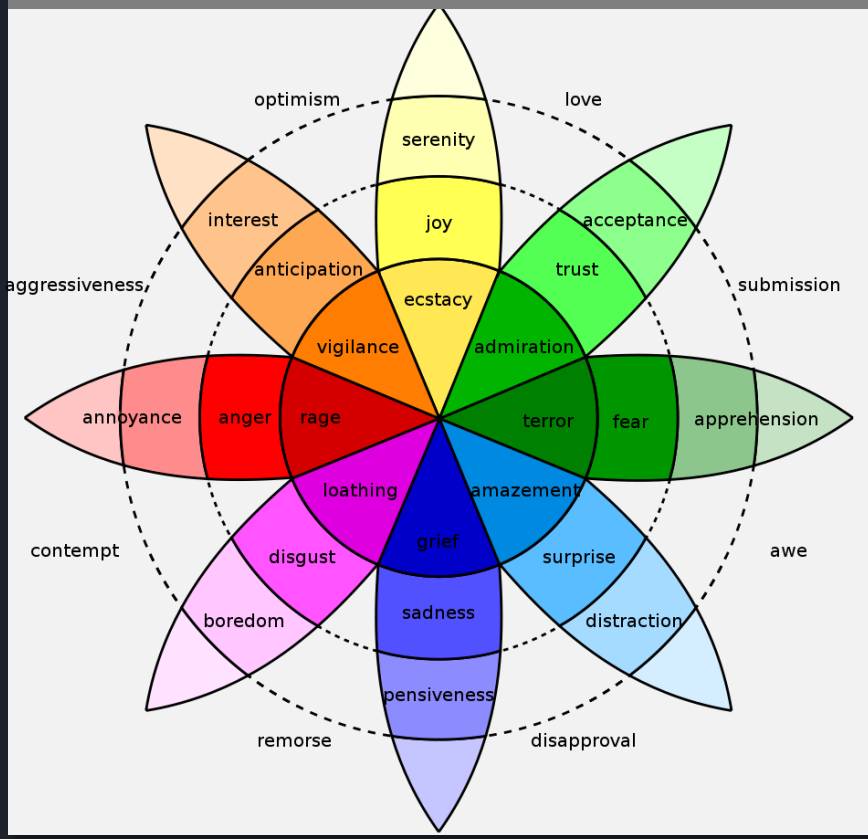
First it looks for the polarized word. Then identifies valence shifters (default 4 words before and two words after) Amplifiers are assigned +.8 and de-amplifiers weight is constrained to -1. Lastly the sum is divided by the square root of the total number of words in the passage.

In reality sentiment is more complex.

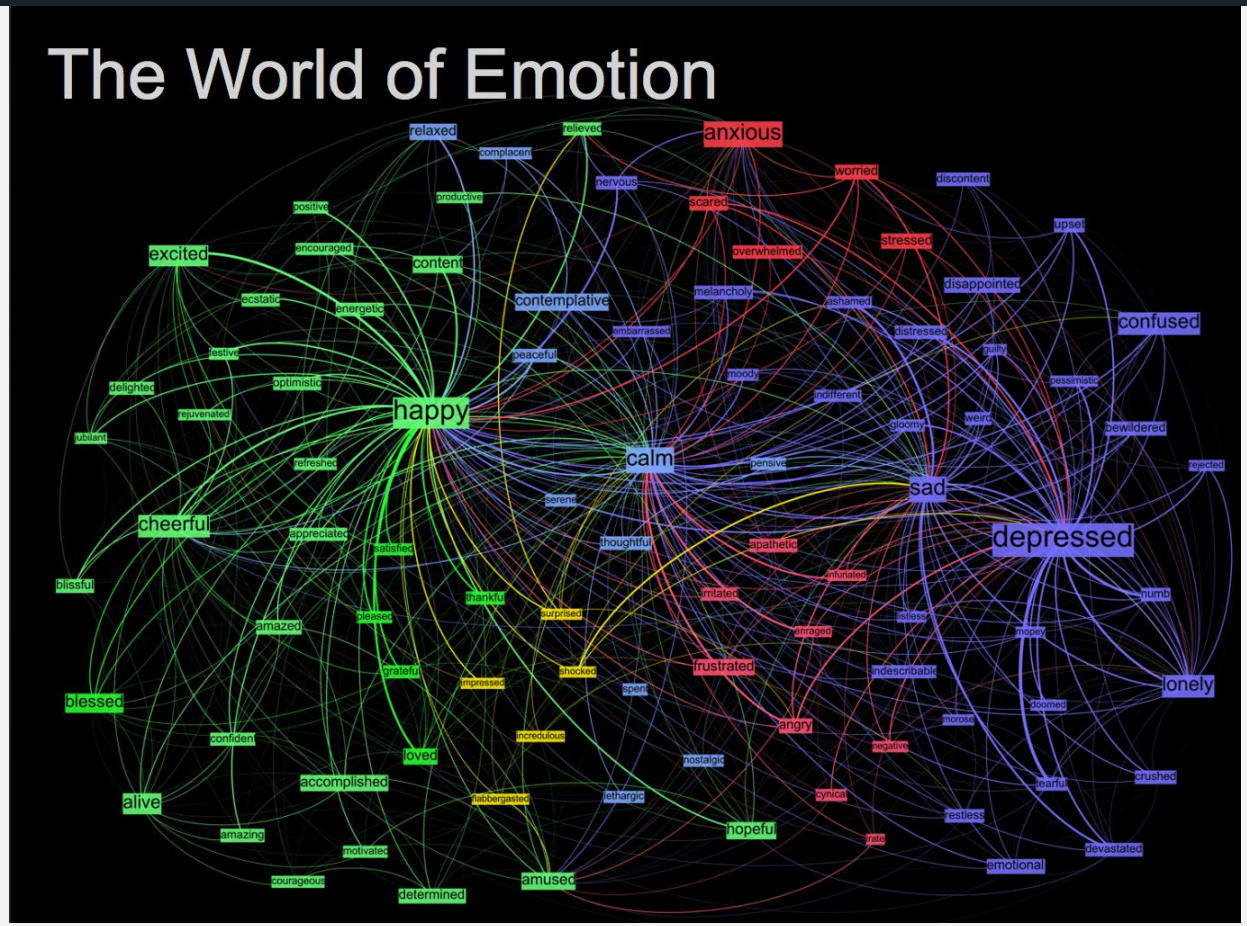
Many Many Emoji



Plutchik's Wheel



Kanjoya's Experience Corpus



Sentiment the Tidy Way!

```
##Tidy Sentiment Analysis
data(sentiments)
sentiments

#Stopwords
data(stop_words)
stop_words

#Add stopwords
custom.stopwords<-data.frame(word=c('amp','beer'),
lexicon='custom')

stop_words<-rbind(stop_words,custom.stopwords)
```

```
> sentiments
# A tibble: 23,165 x 4
          word sentiment lexicon score
         <chr>     <chr>   <chr> <int>
1       abacus     trust     nrc    NA
2      abandon      fear     nrc    NA
3      abandon  negative     nrc    NA
4      abandon   sadness     nrc    NA
5    abandoned     anger     nrc    NA
6    abandoned      fear     nrc    NA
7    abandoned  negative     nrc    NA
8    abandoned   sadness     nrc    NA
9  abandonment     anger     nrc    NA
10 abandonment      fear     nrc    NA
# ... with 23,155 more rows
```

```
> stop_words
# A tibble: 1,151 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
# ... with 1,141 more rows
```

6_TidyText_Sentiment.R

```
> library(tidytext)
> data("sentiments")
> sentiments
# A tibble: 27,314 x 4
          word sentiment lexicon score
         <chr>     <chr>   <chr> <int>
 1       abacus     trust     nrc    NA
 2      abandon      fear     nrc    NA
 3      abandon  negative     nrc    NA
 4      abandon   sadness     nrc    NA
 5    abandoned     anger     nrc    NA
 6    abandoned      fear     nrc    NA
 7    abandoned  negative     nrc    NA
 8    abandoned   sadness     nrc    NA
 9 abandonment     anger     nrc    NA
10 abandonment      fear     nrc    NA
# ... with 27,304 more rows
```

```
> 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
# ... with 1,139 more rows
```

Tidytext is part of the tidy universe including ggplot and dplyr. Code is structured so it is more easily read using the %>%. The data format is a tibble and is in "tidy" format (long form).

```
mtcars %>% group_by(cyl) %>% mutate(rank = min_rank(desc(mpg)))
```

This reads as "Using the mtcars object then group by the cyl vector then mutate a new variable called rank.


Ceci n'est pas une pipe.

6_TidyText_Sentiment.R

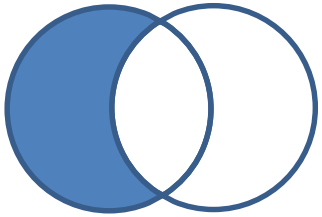
The pipe operator

%>%

Forwards an object so the code is easy to understand & concise.

```
all.tidy <- all.tidy %>%
  anti_join(stop_words)
```

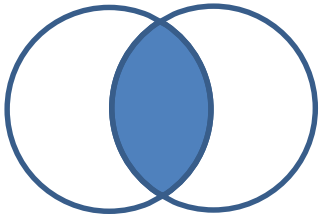
Tweet words Stop words



anti_join()

```
all.sentiment <- all.tidy %>%
  inner_join(nrc.lexicon) %>%
  count(tweet,sentiment) %>%
  spread(tweet, n, fill = 0)
```

text Sentiment words



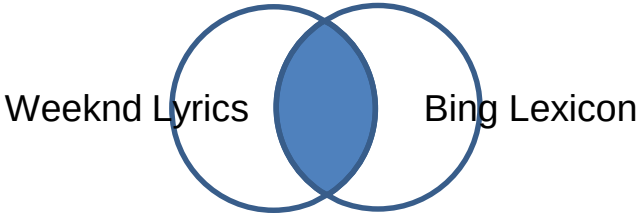
inner_join()

6_TidyText_Sentiment.R

```
# DTM
txtDTM<-DocumentTermMatrix(txtCorpus)
txtDTM
dim(txtDTM)

# Tidy
tidyCorp<-tidy(txtDTM)
tidyCorp
dim(tidyCorp)

# Get bing lexicon
# "afinn", "bing", "nrc", "loughran"
bing<-get_sentiments(lexicon = c("bing"))
head(bing)

# Perform Inner Join
bingSent<-inner_join(tidyCorp,bing,
by=c('term'='word'))
```

DTM is from the “tm” library

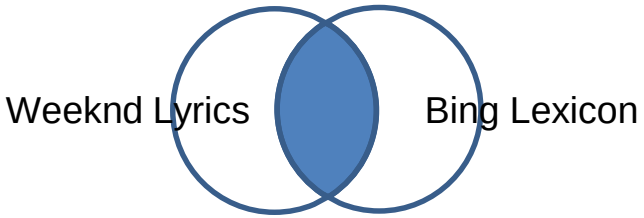
Easy way to make it into a tibble.



6_TidyText_Sentiment.R

```
> table(bingSent$sentiment)

negative positive
      11        4
>
> # Compare with Polarity
> polarity(text)
  all total.sentences total.words ave.polarity sd.polarity stan.mean.polarity
1 all               1         383       -0.409          NA                 NA
```

The polarity function from qdap and the inner_join show similar negative results.



6_TidyText_Sentiment.R

The sentiments tibble has multiple lexicons. Each can be used in an inner join to get different ways of assessing sentiment.

AFINN- Dutch researcher
Words scored -5 to 5

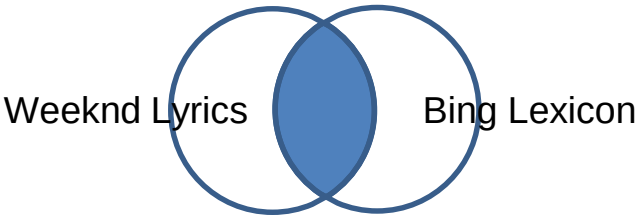
```
> head(afinn)
# A tibble: 6 x 2
        word score
       <chr> <int>
1    abandon    -2
2  abandoned    -2
3   abandons    -2
4   abducted    -2
5  abduction    -2
6 abductions    -2
```

Bing- UofI-Chi Researcher
Words scored Pos/Neg

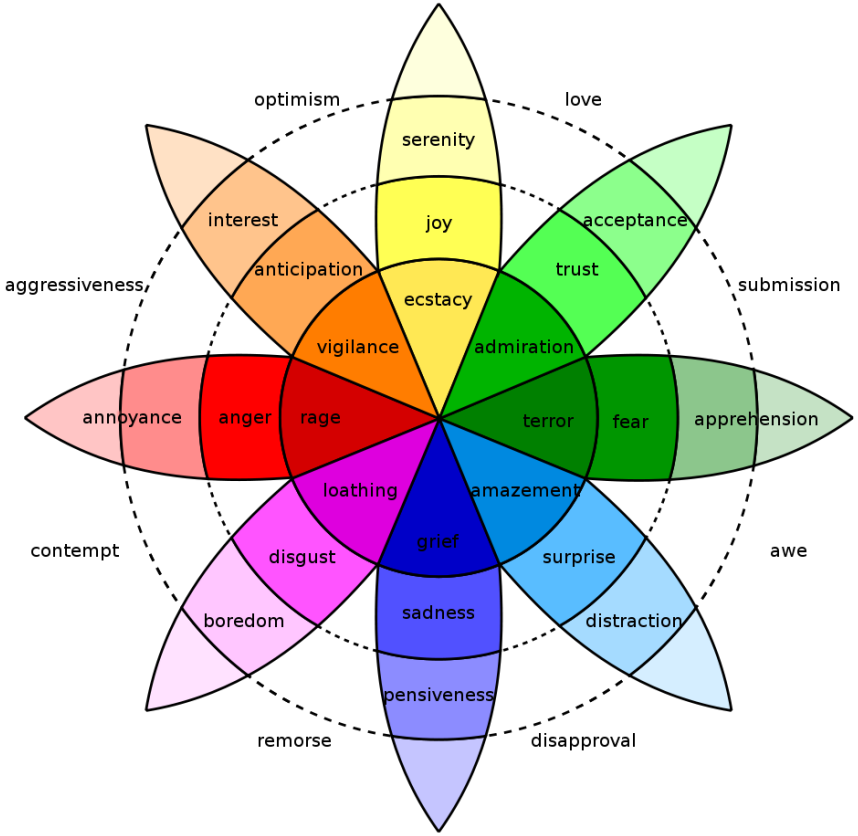
```
> head(bing)
# A tibble: 6 x 2
        word sentiment
       <chr>     <chr>
1    2-faced  negative
2    2-faces  negative
3         a+  positive
4   abnormal  negative
5    abolish  negative
6 abominable  negative
```

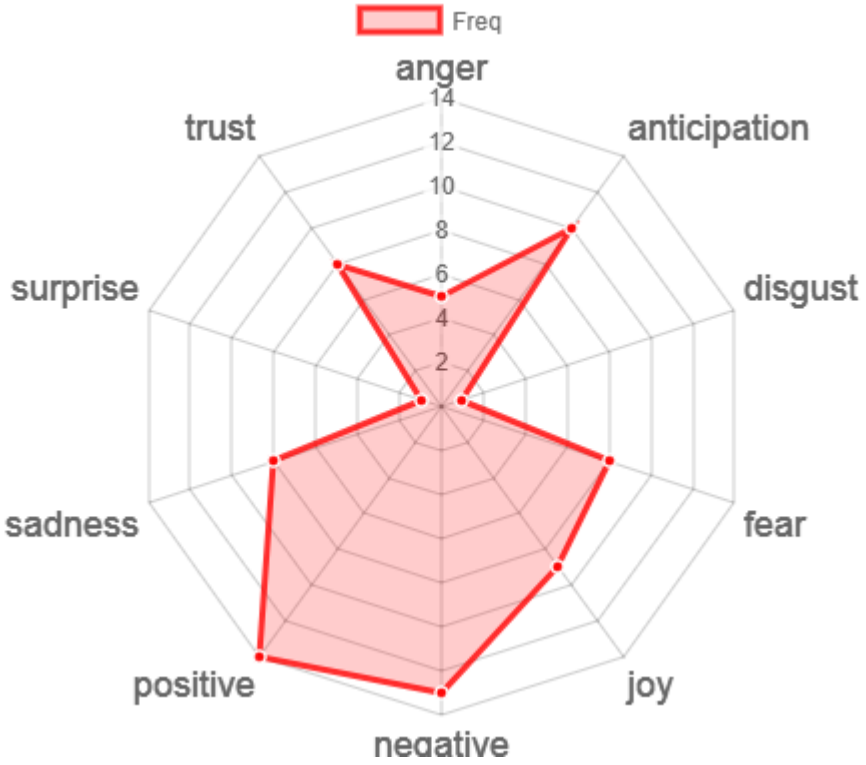
NRC – mTurk
Words classified into 8 primary & pos/neg

```
> head(nrc)
# A tibble: 6 x 2
        word sentiment
       <chr>     <chr>
1     abacus     trust
2    abandon      fear
3    abandon  negative
4    abandon   sadness
5 abandoned     anger
6 abandoned      fear
```



Remember Plutchik's Wheel of emotion? Let's mimic it!





You could remove Pos/Neg before making the radarchart to more closely fit the wheel.

6_TidyText_Sentiment.R

Intro to TM w/R Deep Dive

- ~~What is Text Mining?~~
- ~~String Manipulation & Cleaning~~
- ~~Frequency Counts, Dendrograms & Wordclouds~~
- ~~Sentiment Analysis~~
- Topic Modeling
- OpenNLP/Named Entity Recognition

Goals:

- Learn the basics of text mining
- Apply methods to real data and uses
- Follow a business workflow for stakeholders needing text mining insights

Topic Modeling

LDA

Each document is made up of mini topics.

- Probability is assigned to each document for the specific observed topics.
- A document can have varying probabilities of topics simultaneously.

Technical Explanations:
http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
http://cs.brown.edu/courses/csci2950-p/spring2010/lectures/2010-03-03_santhanam.pdf
***Full disclosure I am not an expert in topic modeling**

Example

Corpus

1.I like to watch basketball and football on TV.
2.I watched basketball and Shark Tank yesterday.
3.Open source analytics software is the best.
4.R is an open source software for analysis.
5.I use R for basketball analytics.

LDA Topics

Topic A: 30% basketball, 20% football, 15% watched, 10% TV…*something to do with watching sports*
Topic B: 40% software, 10% open, 10% source…10% analytics *something to do with open source analytics software*

Documents
1.100% Topic A
2.100% Topic A
3.100% Topic B
4.100% Topic B
5.60% Topic B, 40% Topic A

7_TopicModeling_Sentiment.R

LDA Topic Modeling

library(topicmodels) is not actively maintained. So, library(lda) is used here.

```
txtLex<-lexicalize(txt)
```

Reads raw text, and creates a corpus of n “articles”, and complete word lexicon.

```
txtWordCount<-word.counts(txtLex$documents, txtLex$vocab)
txtDocLength<-document.lengths(txtLex$documents)
```

Word counts for each article
Number of words in each article

```
# LDA Topic Modeling
k <- 5
num.iter <- 25
alpha <- 0.02
eta <- 0.02
set.seed(1234)
fit <- lda.collapsed.gibbs.sampler(documents = txtLex$documents, K = k,vocab = txtLex$vocab,
                                   num.iterations = num.iter,alpha = alpha, eta = eta,
                                   initial = NULL, burnin = 0,compute.log.likelihood = TRUE)
```

Parameters

Model Fit

7_TopicModeling_Sentiment.R

LDA Topic Modeling

One of the cool things about lda is that you can find the prototypical document

```
# Prototypical Document
top.topic.documents(fit$document_sums,1)
```

```
[1] 21 23  2 24  3
```

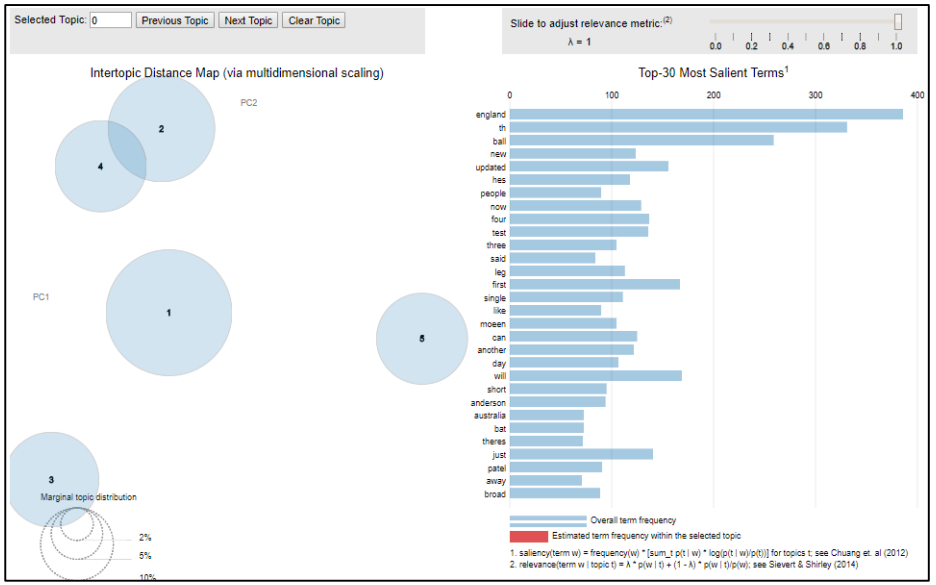
Since K= 5, the 21 article best represents the first topic and so on.

```
> text$webTitle[21]
[1] "A little shared knowledge goes a long way in helping to scale social enterprise"
```

7_TopicModeling_Sentiment.R

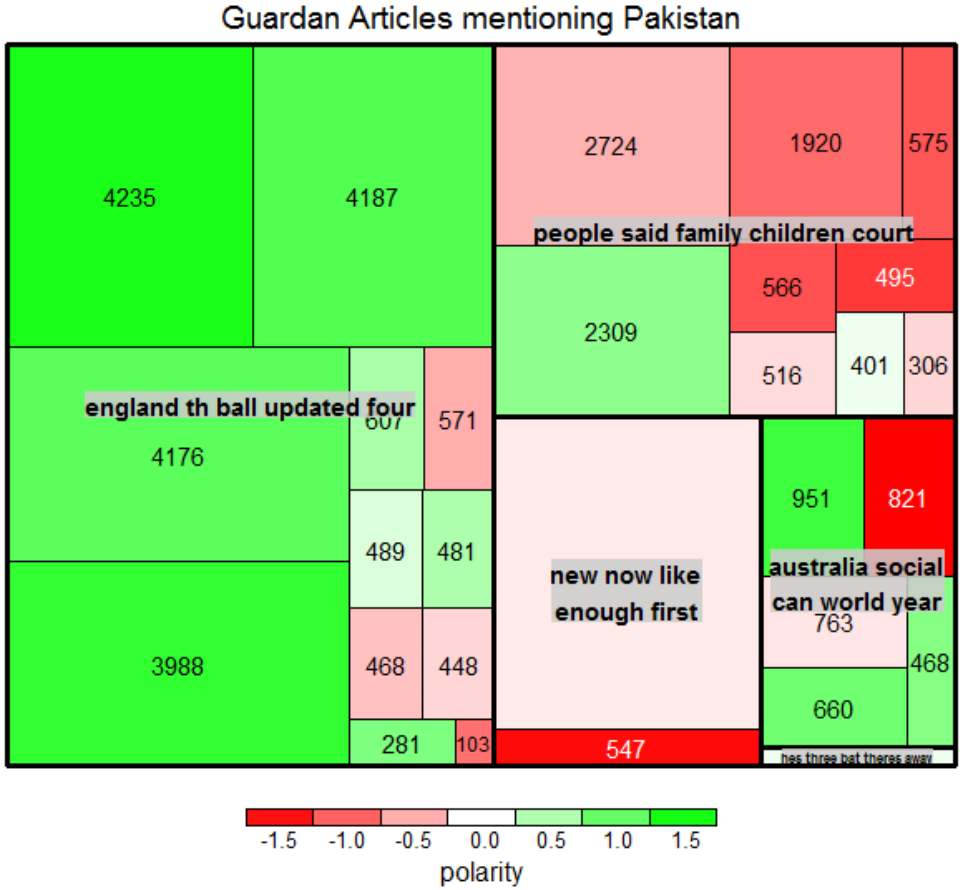
Better yet, let's explore in a dynamic gui w/ ldaVis

```
# LDAvis params
theta <- t(pbapply(fit$document_sums + alpha, 2, function(x) x/sum(x)))
phi <- t(pbapply(t(fit$topics) + eta, 2, function(x) x/sum(x)))

ldaJSON <- createJSON(phi = phi,
                      theta = theta,
                      doc.length = txtDocLength,
                      vocab = txtLex$vocab,
                      term.frequency = as.vector(txtWordCount))

serVis(ldaJSON)
```



7_TopicModeling_Sentiment.R

TREEMAP: multi dimensional representation of the corpus attributes.



- Each article is a small square
- The area of the square is related to the number of terms document length
- Color will represent quick, simple polarity from qdap
- The larger grouping is based on LDA topic assignment

End result is understanding broad topics, their sentiment and amount of the corpus documents devoted to the identified topic.

7_TopicModeling_Sentiment.R

Intro to TM w/R Deep Dive

- ~~What is Text Mining?~~
- ~~String Manipulation & Cleaning~~
- ~~Frequency Counts, Dendrograms & Wordclouds~~
- ~~Sentiment Analysis~~
- ~~Topic Modeling~~
- OpenNLP/Named Entity Recognition

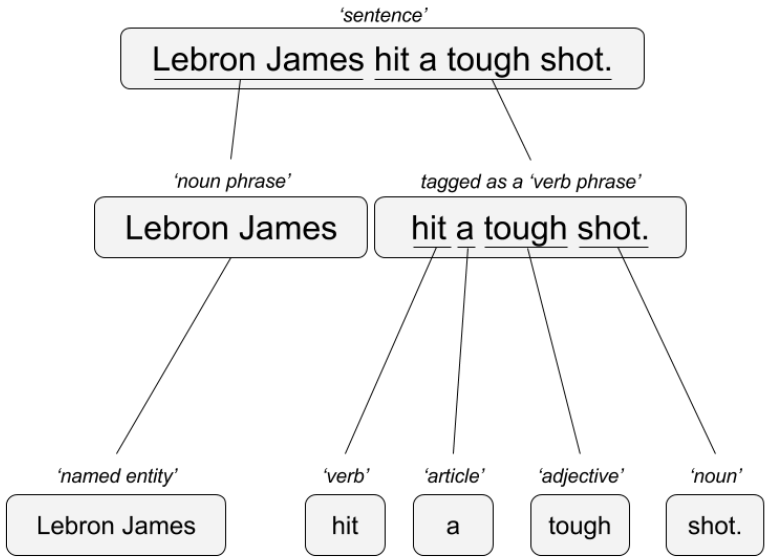
Goals:

- Learn the basics of text mining
- Apply methods to real data and uses
- Follow a business workflow for stakeholders needing text mining insights

Remember this? Text Mining Approaches

"Lebron James hit a tough shot."

Semantic Using Syntactic Parsing



Bag of Words

Named Entity Recognition using OpenNLP

Welcome to Apache OpenNLP

The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text.

Get started Download

R's OpenNLP package wraps the Apache OpenNLP project. However, documentation & examples can be hard to come by.

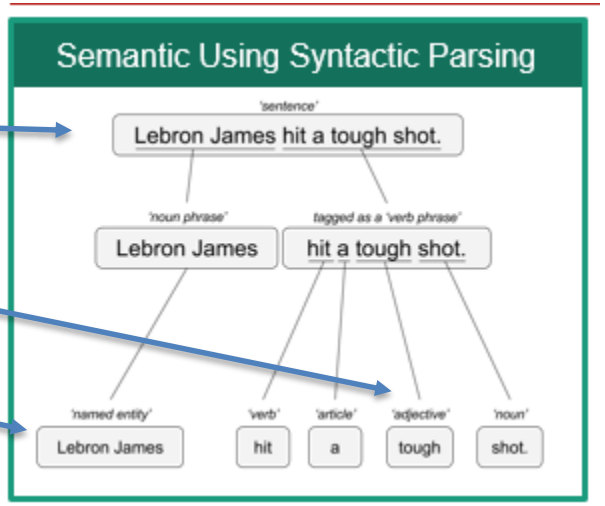
8_Open_Language_Processing.R

library(openNLP)

- https://rpubs.com/lmullen/nlp-chapter
- Uses Annotators to identify the specific item in the corpus. Then holds all annotations in a plain text document. Think of auto-tagging words in a document and saving the document terms paired with the tag.
- Documentation, examples are hard to come by!

Annotations

- Grammatical or POS (Part of Speech) Tagging
- Sentence Tagging
- Word Tagging
- Named Entity Recognition
    - Persons
    - Locations
    - Organizations



8_Open_Language_Processing.R

Annotations Specified

```
#OpenNLP Annotators
persons <- Maxent_Entity_Annotator(kind = 'person')
locations <- Maxent_Entity_Annotator(kind = 'location')
organizations <- Maxent_Entity_Annotator(kind = 'organization')
sent.token.annotator <- Maxent_Sent_Token_Annotator(language = "en")
word.token.annotator <- Maxent_Word_Token_Annotator(language = "en")
pos.tag.annotator <- Maxent_POS_Tag_Annotator(language = "en")
```

Annotations Applied to Text

```
#annotate text
annotations <- annotate(text.s,list(sent.token.annotator,word.token.annotator,pos
.tag.annotator,persons,locations,organizations))
```

Annotate is used in ggplot2 SO you must either change the above to NLP::annotate(…) or ensure ggplot2 is not loaded in your environment!

8_Open_Language_Processing.R

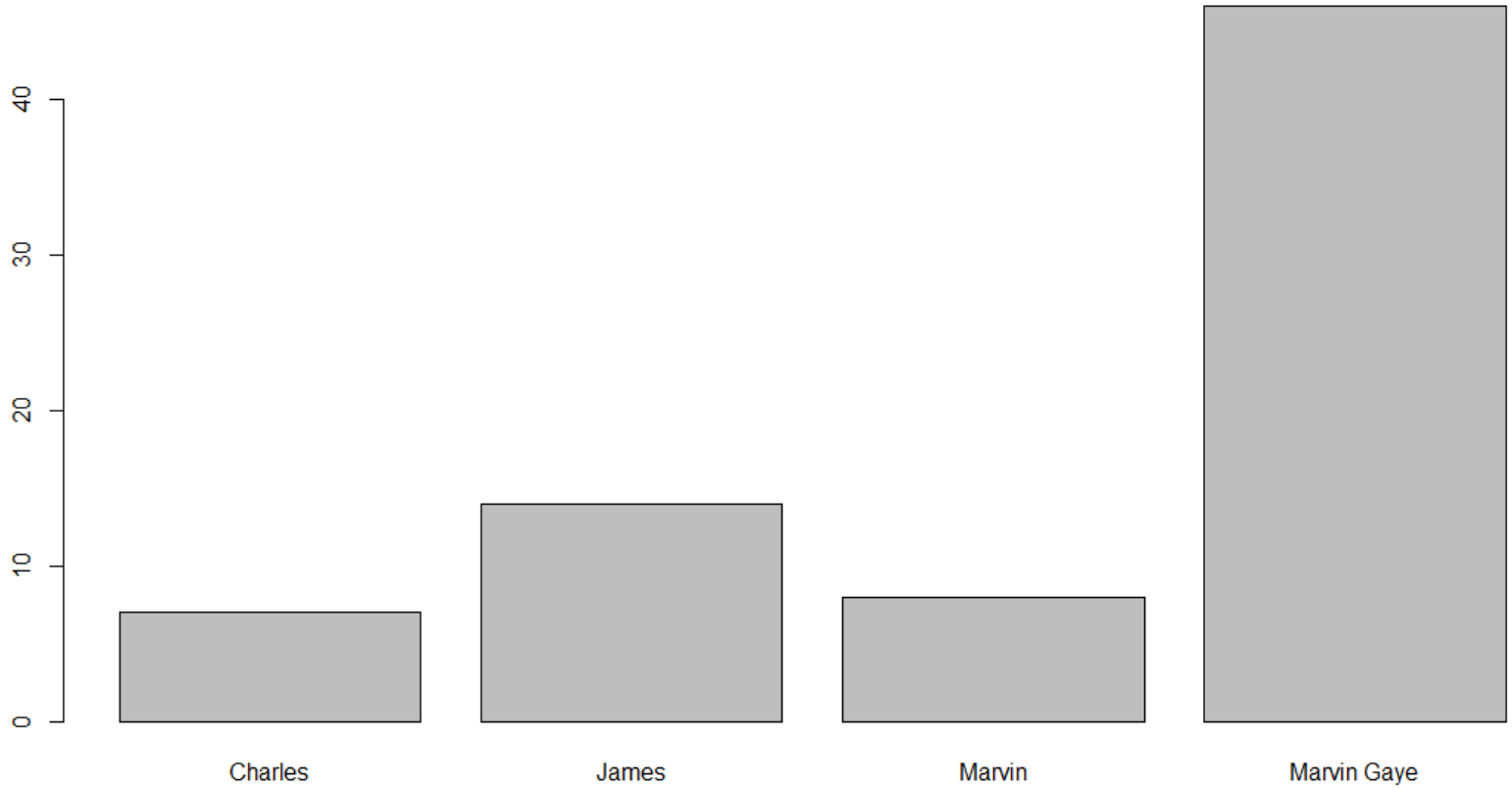
```
#Extract Entities
entities <- function(doc, kind) {
  s <- doc$content
  a <- annotations(doc)[[1]]
  if(hasArg(kind)) {
    k <- sapply(a$features, `[[`, "kind")
    s[a[k == kind]]
  } else {
    s[a[a$type == "entity"]]
  }
}
```

The annotated plain text object is large with a complex structure. This function allows us to extract the tokens by tag kind.

8_Open_Language_Processing.R

The function creates a vector of people, locations and orgs that were “recognized”

```
> head(people)
[1] "Marvin Gaye"   "Marvin"        "LeedsVsSydney" "Marvin Gaye"   "Marvin Gaye"
[6] "Marvin"
> peopleTable<-table(as.factor(people))
> topPeeps<-peopleTable[peopleTable>5]
> barplot(topPeeps)
```



8_Open_Language_Processing.R

Workshop Agenda

9-10:45	10:45-11	11 – 11:45	11:45 – 12	12-12:40	12:40-12:50	12:50-1
Intro to TM w/R	Break	Gather Text	Break	HTML Widgets & Flexdashboard	Optional Break	Scheduling

Goals:

- Learn the basics of text mining
- Apply methods to real data
- Follow a business workflow for stakeholders needing text mining insights

Review: read.csv()

To read a single “comma separated value” file (.csv):

```
text<-read.csv('coffee.csv', header=TRUE)
```

Part of utils package installed with base R. Used for tabled data with a “delimited” to create a data.table.

Example: 1_Keyword_Scanning.R

Some of the alternatives include:

Package::Function	Description
`data.table::fread()`	Fast And Friendly File Finagler
`readr::read_csv()`	Read A Delimited File (Including Csv & Tsv) Into A Tibble
`ff::read.csv.ffdf()`	“Imports csv files into ff data frames (memory-efficient storage of large data on disk and fast access functions)
`sqldf::read.csv.sql()`	Read File Filtered By SQL statement
`bigmemory::read.big.matrix()`	Package for working with “massive matrices” File Interface For A ``Big.Matrix'' class

Review: read.csv() with Multiple Files*

Start with a vector of .csv files OR you can use

```
txtFiles<-c('chardonnay.csv','coffee.csv','beer.csv')
```

OR if you have a folder with many

```
txtFiles<-list.files(pattern = "*.csv")
```

Example: 5_Other_Wordclouds.R

A For loop will read each file individually and add it to the environment.

```
for (i in 1:length(txtFiles)){
  assign(txtFiles[i], read.csv(txtFiles[i]))
  cat(paste('read completed:',txtFiles[i],'\n'))
}
```

Using lapply or pblapply (personal preference) will put all individual files into a single list

```
all<-pblapply(txtFiles,read.csv)
```

rbindlist() from data.table is an efficient way to collapse the list of documents if needed.

**You can actually use this code to read other file types if you change read.csv*

Plain Text files: readLines()

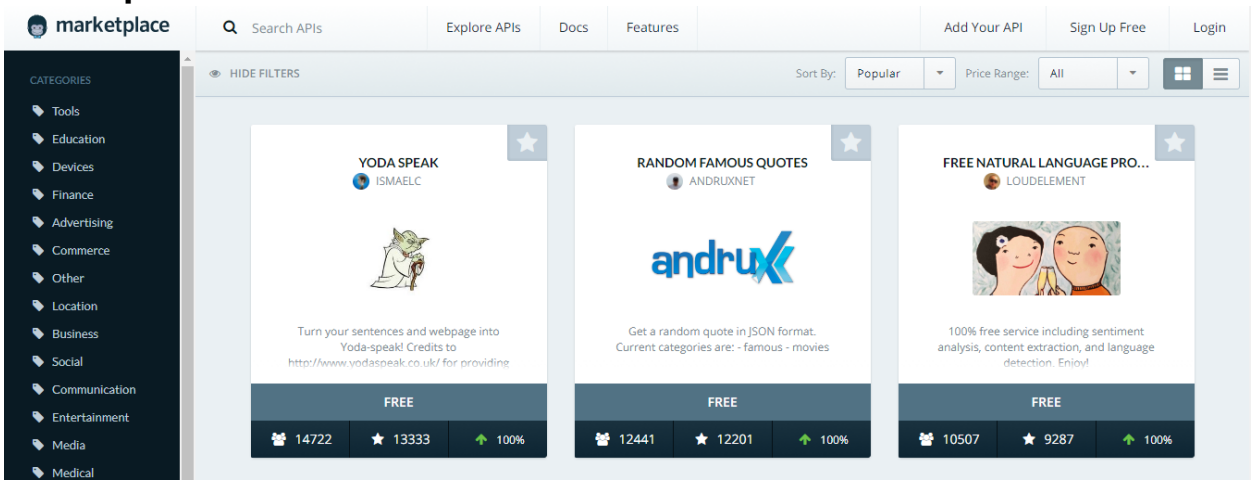
The Weeknd - Starboy (official) ft. Daft Punk

```
# Data
text<-readLines('Weeknd.txt')
```

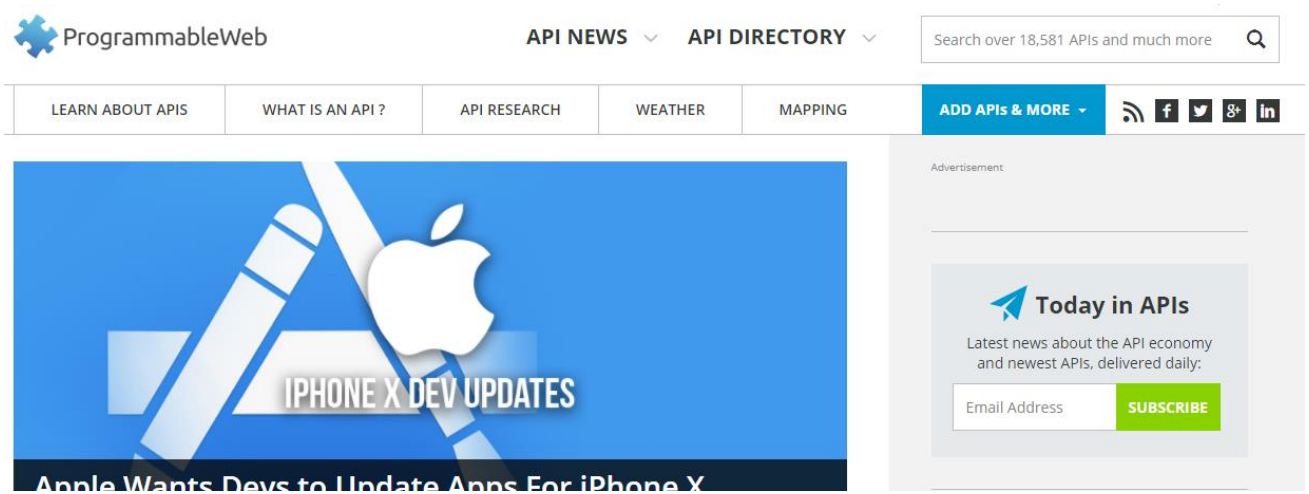
Example: 6_TidyText_Sentiment.R

Now onto APIs!!

Mashape.com



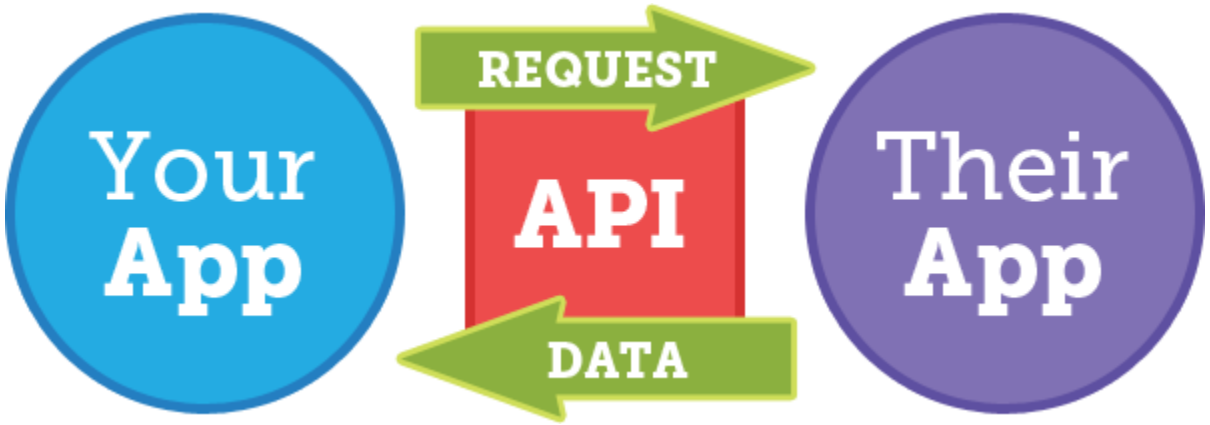
ProgrammableWeb.com



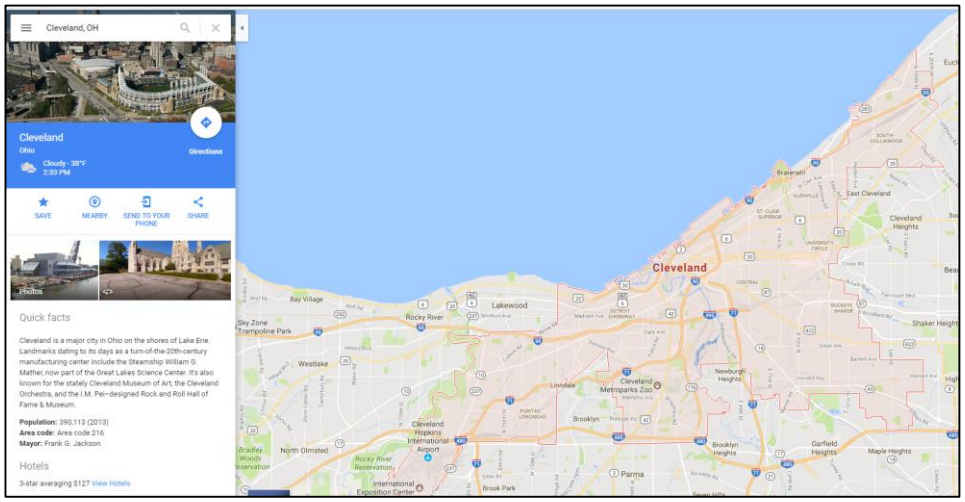
Plus literally millions of others waiting to be found!

Getting Text from an API: What is an API?

Application Program Interface (API)
Clearly defined methods of communication between various software components.



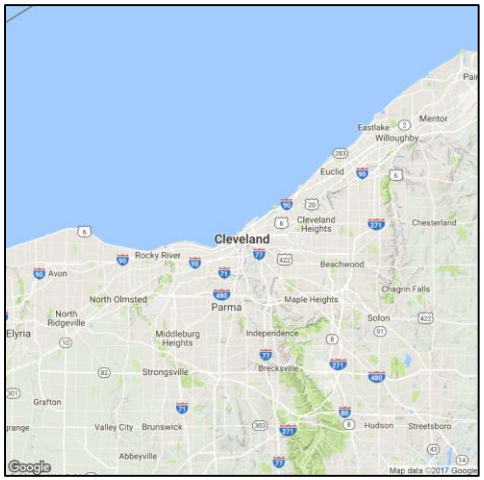
APIs are behind many of the sites you use today.



Google Maps API Services include

- Geocoding (get lat/lon from name & vice versa)
- Base map “tiles”
- Basic Geographic Info


```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "Cleveland",
               "short_name" : "Cleveland",
               "types" : [ "locality", "political" ]
            },
            {
               "long_name" : "Cuyahoga County",
               "short_name" : "Cuyahoga County",
               "types" : [ "administrative_area_level_2", "political" ]
            },
            {
               "long_name" : "Ohio",
               "short_name" : "OH",
               "types" : [ "administrative_area_level_1", "political" ]
            },
            {
               "long_name" : "United States",
               "short_name" : "US",
               "types" : [ "country", "political" ]
            }
         ],
         "formatted_address" : "Cleveland, OH, USA",
         "geometry" : {
```

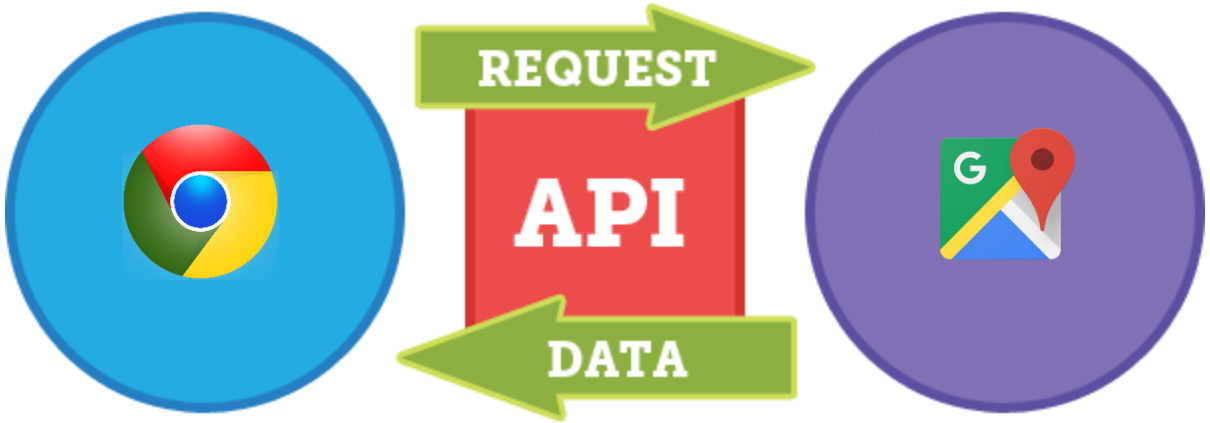
www.google.com/maps/place/Cleveland,+OH/@41.4951143,-81.8462865,11z

maps.googleapis.com/maps/api/staticmap?center=cleveland,+oh&zoom=10&size=640x640&scale=2&maptype=terrain

maps.googleapis.com/maps/api/geocode/json?address=cleveland&sensor=false

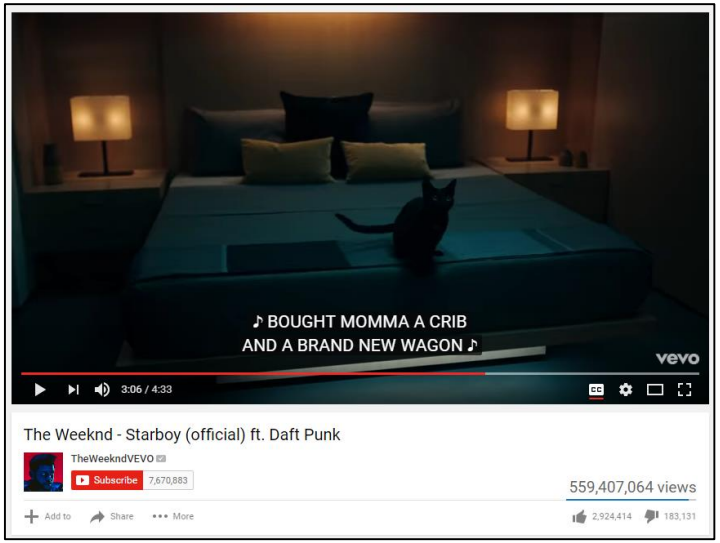
In our diagram, the browser/maps interface…

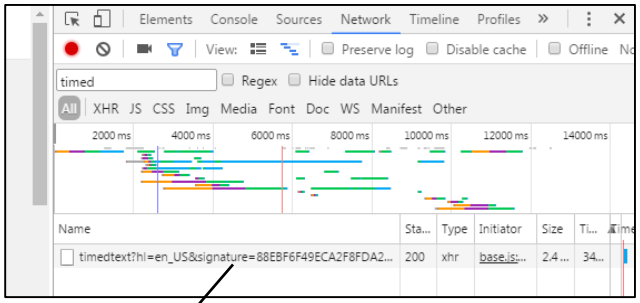
"Your Chrome browser interacts with the maps API services to render the information.

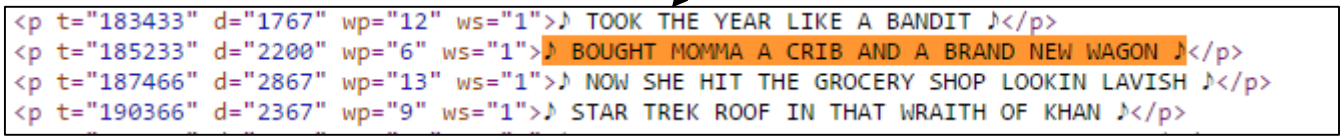


APIs can help unlock information…

Need song lyrics? Access the Closed Caption API file called “timedtext.







https://www.youtube.com/watch?v=34Na4j8AVgA

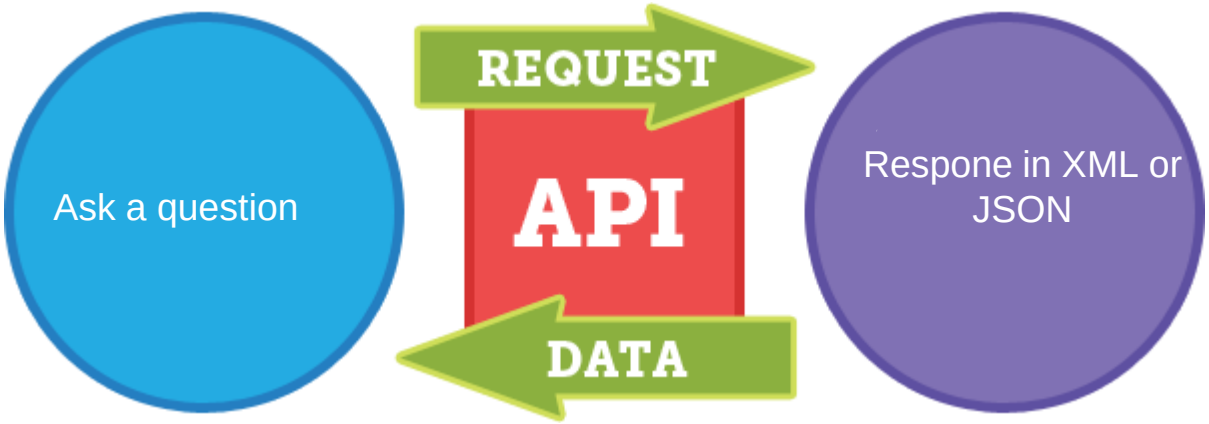
Two popular formats.

XML – Extensible Markup Language

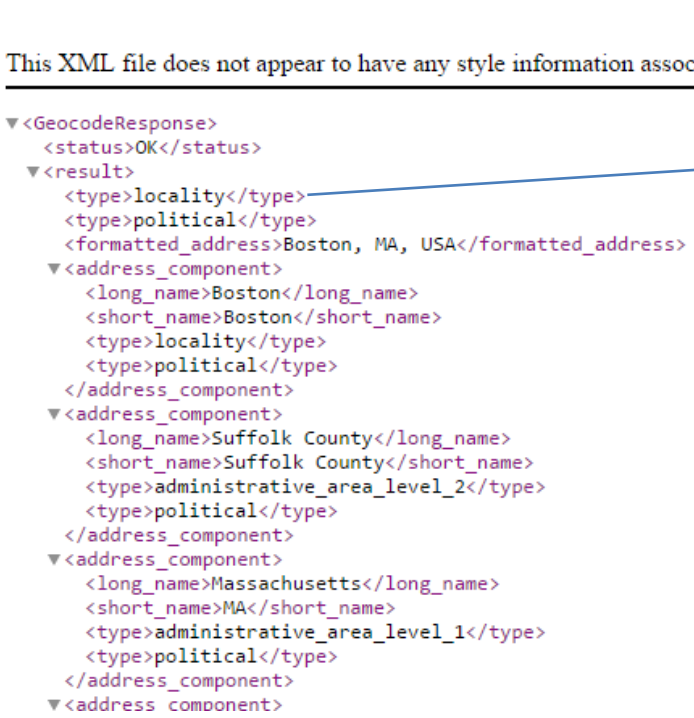
Extensible Markup Language is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
Ever wonder why Excel files went from xls to xlsx? The data is stored as XML.

JSON- Javascript Object Notation

is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types.



Comparing the two formats



http://maps.googleapis.com/maps/api/geocode/**xml**?address=boston&sensor=false

http://maps.googleapis.com/maps/api/geocode/**json**?address=boston&sensor=false

Types of API requests

HTTP GET - Use GET requests to retrieve resource representation/information only e.g. your browsers GETs google map information

HTTP POST/PUT - Use POST APIs to create new subordinate resources e.g. you post a file to kaggle, it is scored and the response is a change in your ranking

HTTP DELETE - DELETE APIs are used to delete resources

HTTP PATCH – PATCH requests are to make partial update on a resource e.g. updating an existing customer record

Generally, a response in the 200s, is a success and anything in the 400s is considerd a failure for various reasons.

Let’s try an API

Request Helpers:

www.Hurl.it

PostMan Chrome Add-in or software

Simple API interactions

- https://newsapi.org/
- https://pipl.com/dev/demo
- http://maps.googleapis.com/maps/api/staticmap?center=Boston+MA&zoom=10&size=640x640

Files to Open

XML Example

https://github.com/kwartler/ODSC/blob/master/p2_Get_Text/GetXML.R

- GetXML.R

JSON Example

https://github.com/kwartler/ODSC/blob/master/p2_Get_Text/instruction_newsAPI.R

- instruction_newsAPI.R

Workshop Agenda

9-10:45	10:45-11	11 – 11:45	11:45 – 12	12-12:40	12:40-12:50	12:50-1
Intro to TM w/R	Break	Gather Text	Break	HTML Widgets & Flexdashboard	Optional Break	Scheduling

Goals:

- Learn the basics of text mining
- Apply methods to real data
- Follow a business workflow for stakeholders needing text mining insights

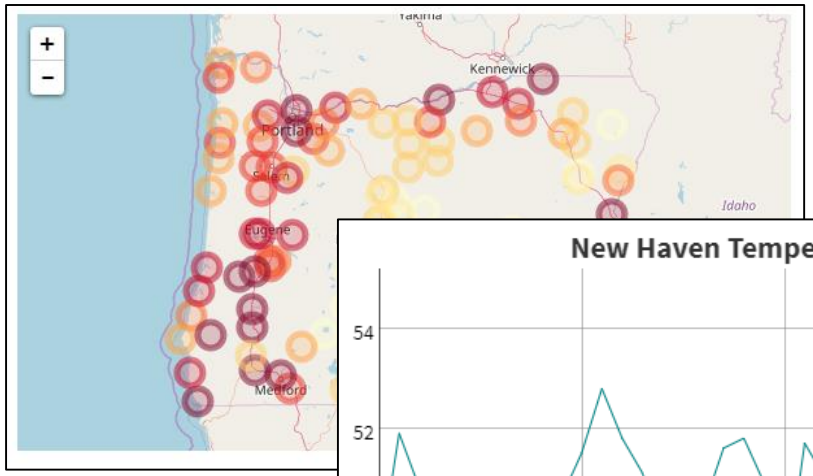
HTML Widgets

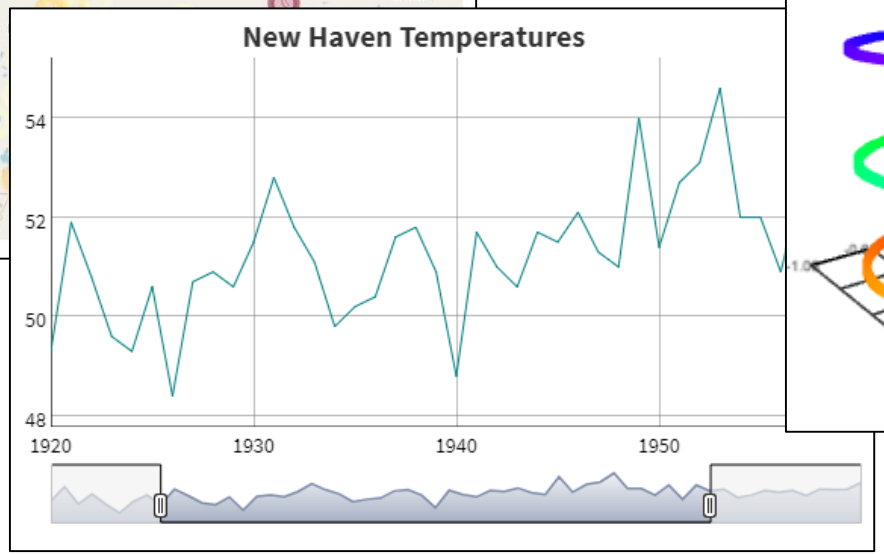
www.HTMLWidgets.org

Let's an R programmer use JavaScript libraries!

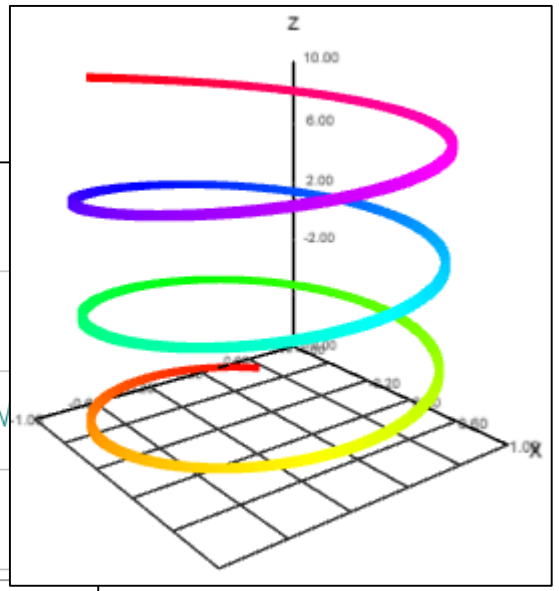
- Dynamic meaning a user can interact with the plot
- Easy to do from within the R console without JS knowledge
- "ships" with all the underlying data, so basic exploration/plotting doesn't need a Shiny or OCPU backend.

http://www.htmlwidgets.org/showcase_leaflet.html





http://www.htmlwidgets.org/showcase_dygraphs.html



http://www.htmlwidgets.org/showcase_threejs.html

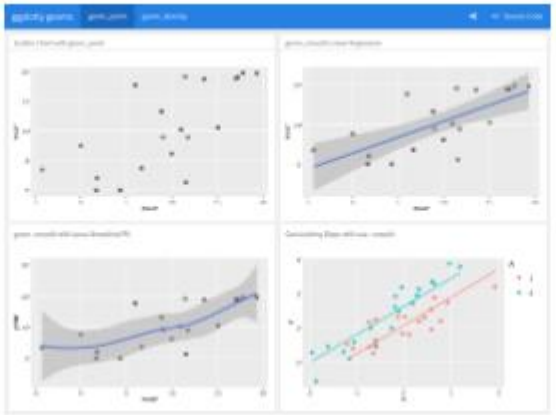
instruction_htmlWidgets.R

FlexDashboard

http://rmarkdown.rstudio.com/flexdashboard/

Let's an R programmer use HTML with Javascript!

- Embed the htmlwidgets, or other plots
- Easy to do from within the R console without HTML knowledge
- Easily change the layout of the dashboard.
- Authoring is done in Rmarkdown (.Rmd) files by "knitting" the components & output is HTML for easy viewing



Open these files:

blank_dashboard.Rmd

top_focal.Rmd

Workshop Agenda

9-10:45	10:45-11	11 – 11:45	11:45 – 12	12-12:40	12:40-12:50	12:50-1
Intro to TM w/R	Break	Gather Text	Break	HTML Widgets & Flexdashboard	Optional Break	Scheduling

Goals:

- Learn the basics of text mining
- Apply methods to real data
- Follow a business workflow for stakeholders needing text mining insights

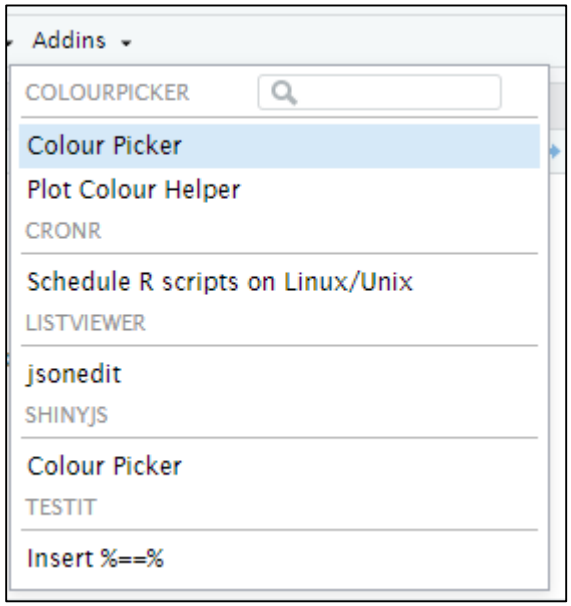
Open these files:

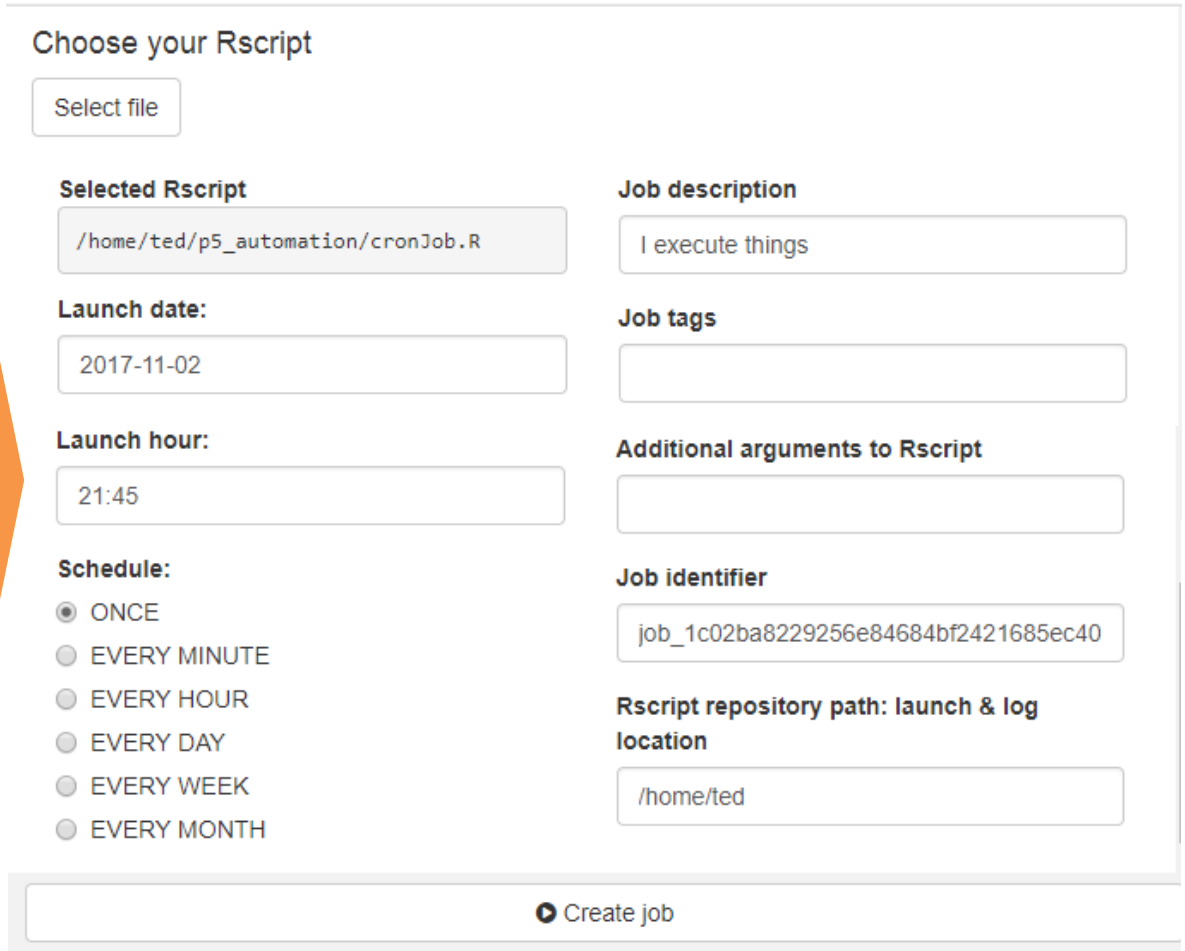
cronDash.Rmd

cronJob.R

Use a scheduler to automate the construction & dissemination of your work!

R Studio cronR Add-In





**Windows task scheduler will be similar.*

Questions?

https://github.com/kwartler/ODSC

@tkwartler

www.linkedin.com/in/edwardkwartler