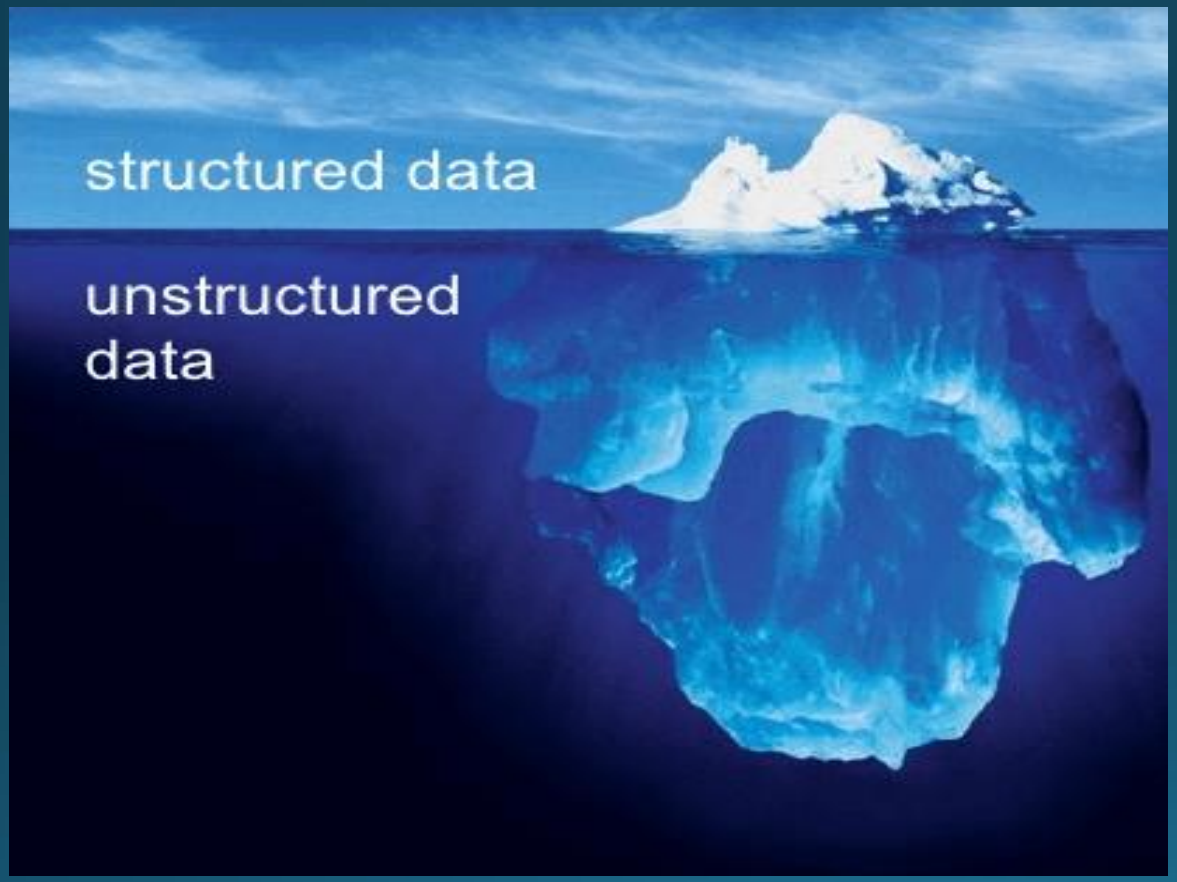
COLUMBIA UNIVERSITY
School of Professional Studies

Applied Analytics: Frameworks and Methods 2

Text Mining

Outline

- Examine the potential of analyzing unstructured data
- Discuss applications of text analysis
- Review various methods used for text analysis
  - Examine process of sentiment analysis
  - Use text as features in a predictive model


structured data
unstructured data

Business Decisions

- Despite the overwhelming amount of unstructured data available, most decisions are based on structured data.

Unstructured Data

- Data that does not fit into rows and columns in a spreadsheet. E.g.,
  - Text
  - Pictures
  - Audio
  - Video …
- We will focus on Text

Text is everywhere

- Books and articles
- Web pages
- News stories
- Call center notes
- Doctor's notes
- Emails
- Reviews
- Tweets
- Social media

And
much of it is generated
by consumers

Text Analysis

- In traditional data analysis projects
  - Data Wrangling takes 80% of the time
  - Analysis and Interpretation takes 20% of the time

- For data analysis projects involving unstructured data
  - Data Wrangling takes 90% of the time
  - Analysis and Interpretation takes 10% of the time

Text Mining: Definition

- *Text mining* is the process of **discovering** and **extracting** meaningful patterns and relationships from text.

Applications of Text Analysis

- Document search (i.e., Google)
- Translation
- Feature Extraction - identifying specific words in text
- Theme extraction – Cluster and Topic Models
- Sentiment analysis
- Classification
- Summarization
- Conversation/Chat

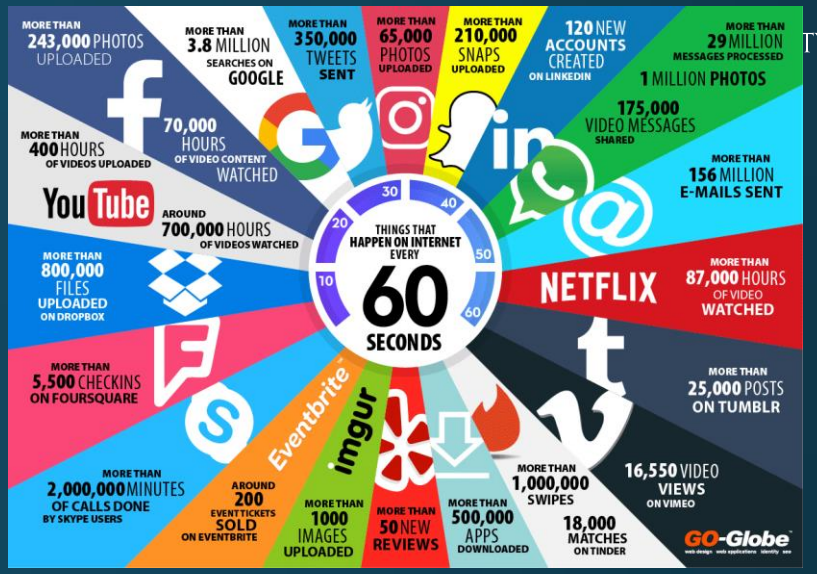
Application - Examples

COLUMBIA UNIVERSITY
School of Professional Studies

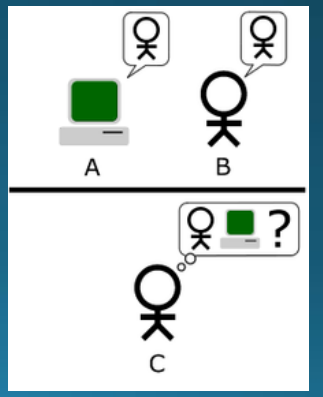
- Spam filtering
- Creating suggestion and recommendations (like amazon)
- Monitoring public opinions (for example in blogs or review sites)
- Customer service, call center and/or email support
- Automatic labeling of documents in business libraries
- Measuring customer preferences by analyzing qualitative interviews
- Fraud detection by investigating notification of claims
- Fighting cyberbullying or cybercrime in IM chat

But, text analysis is hard

- There is a lot of text data
- But, text by people
  - is loosely structured
  - includes typographical errors, abbreviations, emoticons
  - is multilingual
  - includes sarcasm, double meaning, puns, mixed languages (e.g., hinglish, chinglish and spanglish), and abbreviations
- Turing Test
  - Which ones is human?



Source: Go-Globe



Process: Sentiment Analysis

Steps in Process

- Get text
- Explore text
- Prepare text
- Tokenization
- Categorize tokens using a lexicon
- Summarize results

Get Text

- Manually Copy
- Pull from databases
- API
- Scrape

Explore Text

- Explore general characteristics of text
  - Number of letters, words, sentences, urls, emojis
- Identify patterns
  - Define patterns using regular expressions

Prepare Text

Document: typically the unit of analysis

★★★★★ **Fun and informative**

By Michel Baudin on November 13, 2013

Format: Hardcover

This book is less about how to use big data to your advantage than about spotting errors and attempts to deceive. It makes it both fun to read and informative. The author whets your appetite with a prologue about airlines misinterpreting data on flight delays, and then goes on to expose the games deans play to to boost their universities' rankings and make the employment of recent grads look better than it is. He then goes on to debunk claims about obesity and diets in the US, and the measurements of the effects of various marketing tactics.

When he moves on to economic data, on the other hand, he is much less severe with the adjustments and filters applied to statistics on jobs and prices, and explains how the data would be much more misleading without them. I skipped Part IV, about Sporting data. It is largely about fantasy football, which I know nothing about, but the rest of the book was well worth the time I spent on it.

Comment | One person found this helpful. Was this review helpful to you? Yes No Report abuse

Corpus

★★★☆☆ **A decent book with a central theme**

By Chandler Nguyen on May 1, 2016

Format: Kindle Edition | Verified Purchase

This is a decent book on the subject, the author tried to use real life examples to explain different statistics concepts.
I think it is suitable statisticians or analytics team members who dont know how to express the business impact of their output easily. It is quite a long book for a central theme though and I think I understand the key takeaways after 2-3 chapters.

Comment | Was this review helpful to you? Yes No Report abuse

★★★☆☆ **Numbers Rule Your World is Better**

By Mr. Sir on August 9, 2015

Format: Hardcover | Verified Purchase

as with Fung's previous book, this book is interesting and informative. The problem is that it lacked the witty spark and interesting stories of the first one. I'd rather read about CDC methods for tracking down outbreaks than fantasy football gurus any day.

Is it bad? No. It's just not the sensation that Fung's previous book was.

Comment | Was this review helpful to you? Yes No Report abuse

★★★☆☆ **Interesting, but blog-like**

By Joy Thompson on June 25, 2015

Format: Kindle Edition | Verified Purchase

This book reads like a set of blog entries. I've read his other book which I thought was better. I liked this book, it was interesting, but I'd hesitate to recommend it due to the style. I'm a little surprised at all the really good reviews saying this book is well written. I think it could have been a very good book with better editing, but it's rambling and loose and conversation-like. I have higher expectations when I pay for a book.

Comment | Was this review helpful to you? Yes No Report abuse

★★★★☆ **Four Stars**

By Saina on May 31, 2015

Format: Hardcover | Verified Purchase

Good tead for analytics professionals.

Comment | Was this review helpful to you? Yes No Report abuse

★★★★★ **Required reading for those engaged in Data Analysis. Eminently readable and practical. Great examples.**

By David Alan Tussey on September 9, 2014

Format: Hardcover | Verified Purchase

Excellent book to provide practical statistics to the (exaggerated) world of big data. Kaiser Fung has a great way of providing

- Corpus, representing a collection of text documents

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

Prepare Text - Stemming

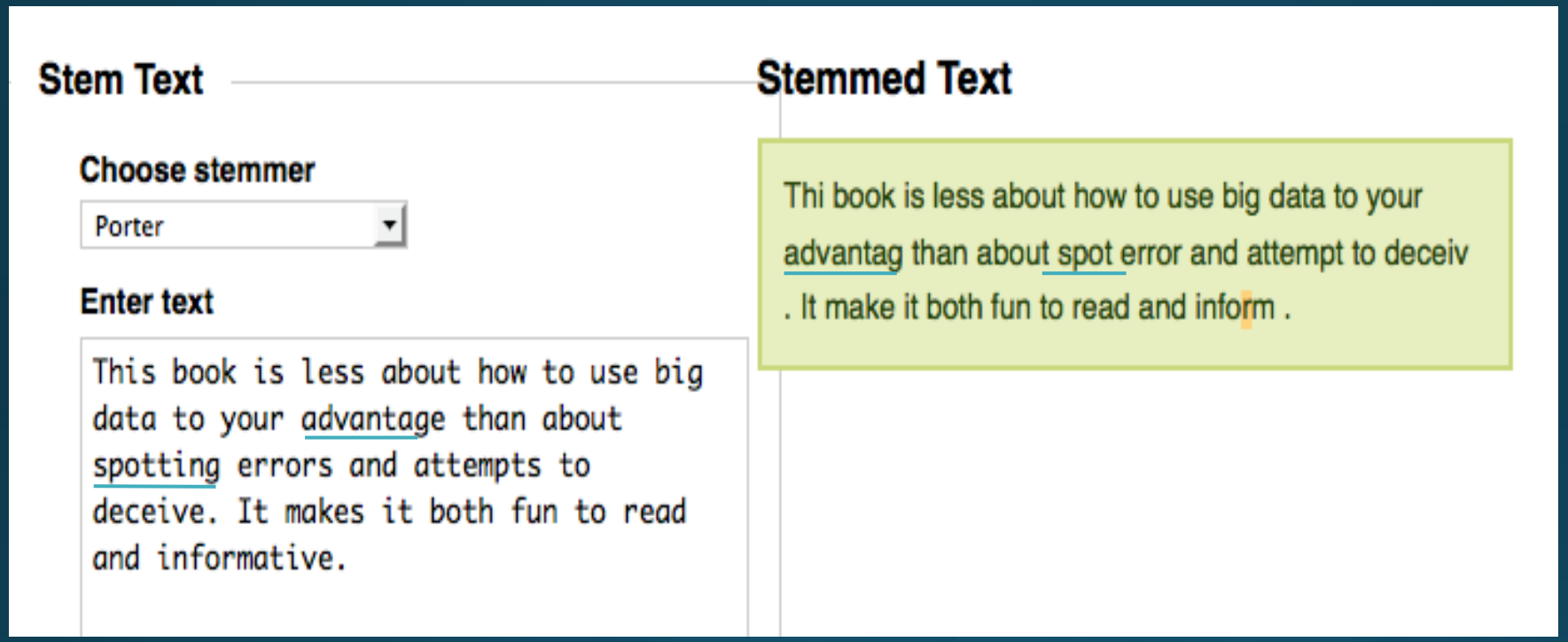
- Retain meaning while ignoring parts of speech, plurality, conjugations, grammar, etc.

E.g.,

	user	engineering
	users	engineered
	used	engineer
	using	
stem:	use	engineer

- **Usefulness**
  - improving effectiveness of retrieval and text mining
    - matching similar words
  - reducing indexing size
    - combing words with same roots may reduce indexing size as much as 40-50%.

Prepare Text - Stemming



advantage >> advantag
spotting >> spot
errors >> error
attempts >> attempt
deceive >> deceiv
informative >> inform

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

Tokenization

COLUMBIA UNIVERSITY
School of Professional Studies

Corpus

A decent book with a central theme
By Chandler Nguyen on May 1, 2016
Format: Kindle Edition | Verified Purchase

This is a decent book on the subject, the author tried to use real life examples to explain different statistics concepts.
I think it is suitable statisticians or analytics team members who dont know how to express the business impact of their output easily. It is quite a long book for a central theme though and I think I understand the key takeaways after 2-3 chapters.

Comment | Was this review helpful to you? Yes No Report abuse

Numbers Rule Your World is Better
By Mr. Sir on August 9, 2015
Format: Hardcover | Verified Purchase

as with Fung's previous book, this book is interesting and informative. The problem is that it lacked the witty spark and interesting stories of the first one. I'd rather read about CDC methods for tracking down outbreaks than fantasy football gurus any day.

Is it bad? No. It's just not the sensation that Fung's previous book was.

Comment | Was this review helpful to you? Yes No Report abuse

Interesting, but blog-like
By Joy Thompson on June 25, 2015
Format: Kindle Edition | Verified Purchase

This book reads like a set of blog entries. I've read his other book which I thought was better. I liked this book, it was interesting, but I'd hesitate to recommend it due to the style. I'm a little surprised at all the really good reviews saying this book is well written. I think it could have been a very good book with better editing, but it's rambling and loose and conversation-like. I have higher expectations when I pay for a book.

Comment | Was this review helpful to you? Yes No Report abuse

Four Stars
By Saina on May 31, 2015
Format: Hardcover | Verified Purchase

Good tead for analytics professionals.

Comment | Was this review helpful to you? Yes No Report abuse

Required reading for those engaged in Data Analysis. Eminently readable and practical. Great examples.
By David Alan Tussey on September 9, 2014
Format: Hardcover | Verified Purchase

Excellent book to provide practical statistics to the (exaggerated) world of big data. Kaiser Fung has a great way of providing

Document: typically the unit of analysis

Fun and informative
By Michel Baudin on November 13, 2013
Format: Hardcover

This book is less about how to use big data to your advantage than about spotting errors and attempts to deceive. It makes it both fun to read and informative. The author whets your appetite with a prologue about airlines misinterpreting data on flight delays, and then goes on to expose the games deans play to to boost their universities' rankings and make the employment of recent grads look better than it is. He then goes on to debunk claims about obesity and diets in the US, and the measurements of the effects of various marketing tactics.

When he moves on to economic data, on the other hand, he is much less severe with the adjustments and filters applied to statistics on jobs and prices, and explains how the data would be much more misleading without them. I skipped Part IV, about Sporting data. It is largely about fantasy football, which I know nothing about, but the rest of the book was well worth the time I spent on it.

Comment | One person found this helpful. Was this review helpful to you? Yes No Report abuse

Token/Term

WordPunctTokenizer

1. This | book | is | less | about | how | to | use | big | data | to | your | advantage | than | about | spotting | errors | and | attempts | to | deceive | .
2. It | makes | it | both | fun | to | read | and | informative | .

Categorize Tokens

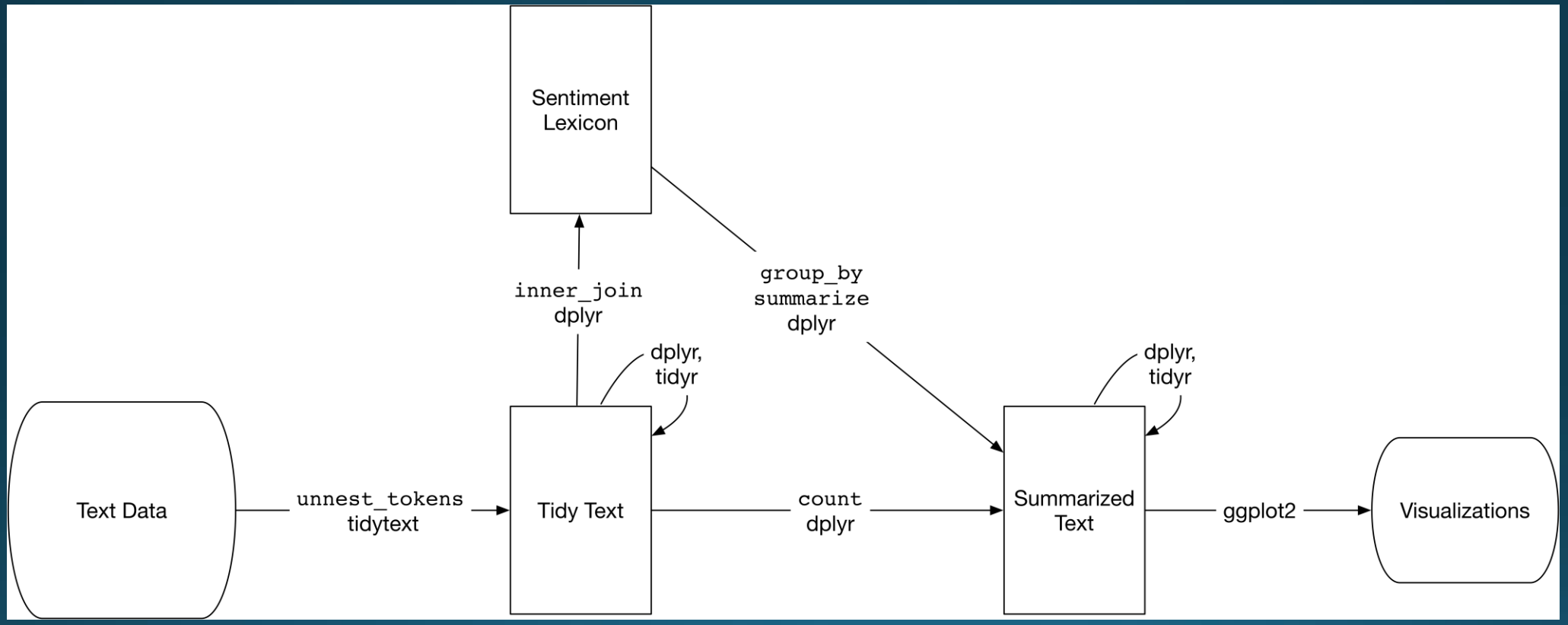
COLUMBIA UNIVERSITY
School of Professional Studies

- Tokens may be grouped using a lexicon.
- A lexicon is a taxonomy of tokens. Choice of lexicon will vary based on the goal of the analysis. Here is a short list of lexicons with an open license
  - Binary (positive/negative) Sentiment: categorizes words in a binary fashion as positive or negative. Lexicons include nrc, emoji sentiment, bing,
  - Emotion: Categorizes words based on the emotion conveyed. Lexicons include nrc
  - Sentiment score: Scores words based on sentiment. afinn, Jockers Polarity table, Jockers Sentiment, Senticnet, SentiWordNet, Slang phrases, SOCal,
  - Domain-specific lexicon: Used to code words with domain-specific jargon. E.g., loughran (scores words relevant to Finance), accounting, entrepreneurship, market orientation, etc.
  - Interpreting Emojis and emoticons: List and associated emotions
  - Common text filters: Lists of words to be removed before processing. E.g., Common names (based on 1990 US census), profane words, stopword lists
  - Other: valence shifters, contraction conversions, clichés, POS (pronouns, prepositions, interjections)

Summarize

- Descriptive
  - Sentiment score
  - Distribution of categories of emotions expressed
- Visualize

Process using library (tidytext)



Other Techniques for Text Summarization

- Topic modelling
  - Useful for discovering underlying themes or topics
  - Two common topics models
    - Latent Dirichlet Allocation
    - Correlated Topic Model
- Latent semantic analysis
  - Used to understand the meaning of a collection of documents
  - Relies on singular value decomposition
- Text clustering and document clustering
- Other machine learning driven text summarization

R Illustration – Data Exploration & Sentiment Analysis

Explore Text & Conduct Sentiment Analysis

Get Data

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 in Reviews

You must read the data before trying to run 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)
```

Explore Text

Structure of dataset

```
str(videogame)
```

```
## 'data.frame':    26652 obs. of  3 variables:
##  $ id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ review       : chr  "1st shipment received a book instead of the game.2nd shipment got a FAKE one. Game
arrived with a wrong key ins"| __truncated__ "I still haven't figured this one out. Did everything as instruct
ed but the game never installed. Strange. Since"| __truncated__ "I bought this and the key didn't work.  It wa
s a gift, and the recipient wasn't able to solve the problem.  It "| __truncated__ "I love it! Use it all the
time. Really works perfectly with all games that you need a mic for." ...
##  $ review_rating: int  1 2 1 5 5 1 5 5 5 5 ...
```

- number of reviews
  nrow(videogame)

Ratings of Reviews


```
median(videogame$review_rating) # median review rating
```

```
## [1] 5
```

```
mean(videogame$review_rating) # mean review rating
```

```
## [1] 4.194544
```

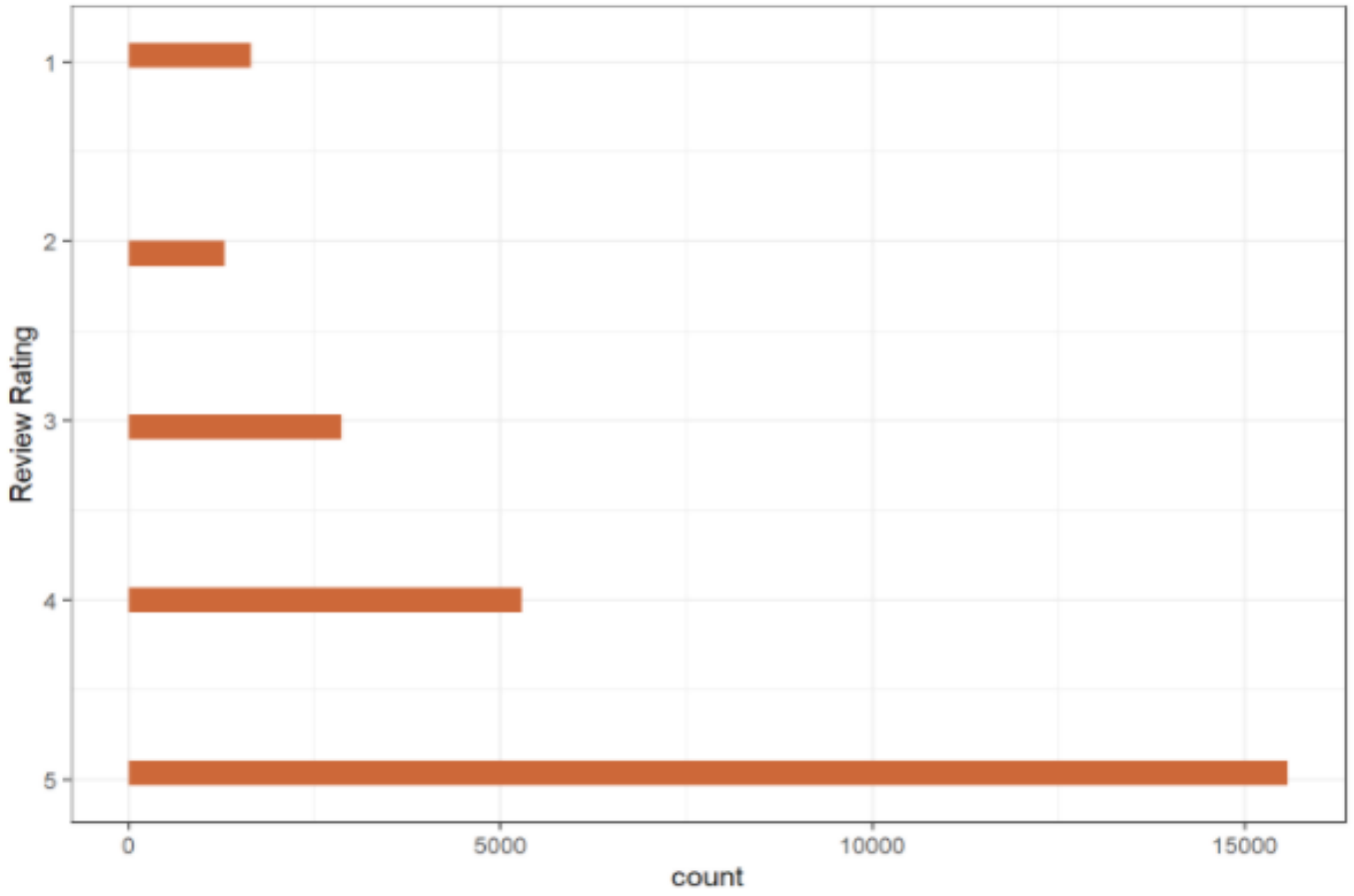
```
# Using dplyr
library(dplyr)
videogame%>%
  summarize(average_rating = mean(review_rating),
            median_rating = median(review_rating))
```

average_rating <dbl>	median_rating <dbl>
4.194544	5

- What do the median and average review rating tell you ?

Distribution of Reviews

```
library(ggplot2); library(ggthemes)
ggplot(data=videogame,aes(x=review_rating))+
  geom_histogram(fill='sienna3')+
  theme_bw()+
  scale_x_reverse()+
  xlab('Review Rating')+
  coord_flip()
```



- Here we visualize the number of reviews with a particular review rating.

Review 617

To get a feel of the text of the reviews, let us examine simple text characteristics of a randomly selected review, 617.

Characters

One measure of length of a review is the number of characters.

```
videogame$review[617]
```

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

```
nchar(videogame$review[617])
```

```
## [1] 717
```

nchar() includes space as character.

Words

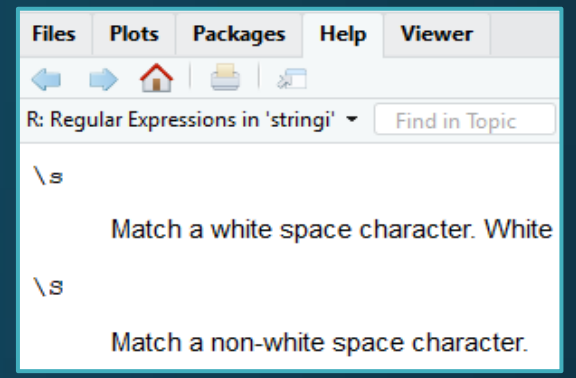
Another measure of length is number of words. While counting words may seem too elementary to warrant discussion, take a moment to think of how you would tell a computer to identify a word in text.

In the illustration below, we will convey the definition of a word to the computer as a pattern. The function str_count from the library stringr will identify words based on the pattern and count them.

In this case, it will count the number of times non white space appears in review 617. More about patterns below.

```
library(stringr)
str_count(string = videogame$review[617],pattern = '\\S+')
```

```
## [1] 127
```




```
> str_count(string="Columbia University",pattern='\\s+')
[1] 1
> str_count(string="Columbia University",pattern='\\S+')
[1] 2
```

Sentences

Now, let us count the sentences. As above, it is worth pondering the definition of a sentence. The definition of a sentence is then encoded as a regular expression. Regular expressions (regex) is a framework for teaching a computer how to recognize patterns of text. Regex has a consistent implementation across many different programming languages and you can read more about it here.

We will define a sentence as being as "a set of characters or punctuation (comma, quote) or spaces that end with one or more period, question mark, exclamation mark or combination of them." The following regular expression would match this definition:

```
str_count(string = videogame$review[617],pattern = "[A-Za-z,;'\"\\s]+[^.!?]*[.?!]")
```

```
## [1] 12
```

All Reviews

Number of characters, words or sentences in review 617 is hard to evaluate out of context. So, let us get some context by examining a summary of characters, words, and sentences across all reviews.

Characters

Number of characters, words or sentences in review 617 is hard to evaluate out of context. So, let us get some context by finding the mean and median across all reviews. Characters across all reviews.

```
summary(nchar(videogame$review))
```

average number of characters in a review

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     3.0   142.0   270.0   673.4   667.0 24012.0
```

Words

```
summary(str_count(string = videogame$review,pattern = '\\S+'))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0    27.0    51.0   123.3   124.0  4467.0
```

Sentences

```
summary(str_count(string = videogame$review,pattern = "[A-Za-z,;'\"\\s]+[^.!?]*[.?!]"))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   4.000   7.575   8.000 416.000
```

In summary,

```
library(dplyr)
videogame %>%
  select(review)%>%
  mutate(characters = nchar(review),
         words = str_count(review,pattern='\\S+'),
         sentences = str_count(review,pattern="[A-Za-z,;'\"\\s]+[^.!?]*[.?!]"))%>%
  summarize_at(c('characters','words','sentences'),.funs = mean,na.rm=T)
```

characters	words	sentences
<dbl>	<dbl>	<dbl>
673.4075	123.3287	7.575116

1 row

- In this example, we define the longest and shortest review as the review with the most and least number of words respectively

Shortest Review

Longest Review

How many words are in the shortest review?

Longest and Shortest Review

Let's use the number of words in a review to measure the length of a review. The shortest review

```
shortest_review_index = which.min(str_count(string = videogame$review,pattern = '\\S+'))
videogame$review[shortest_review_index]
```

```
## [1] "lol"
```

And the longest review

```
longest_review_index = which.max(str_count(string = videogame$review,pattern = '\\S+'))
videogame$review[longest_review_index]
```

```
## [1] "You all knew this was coming, buckle in too... it's gonna be a wild ride.Bravely Default, as many of you know, is Square-Enix's latest RPG. I like to think of it as Square holding this game out in front of you going &#34;Hey, look LOOK We can still make RPGs... no no don't look at FFXIII look at this.&#34; This is of course not true, but it's funny in my head. The game is Final Fantasy down
```

How many words are in the shortest review

```
str_count(videogame$review[shortest_review_index],pattern = '\\S+')
```

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

Correlation between Length and Rating

The variation in the length of reviews is striking. This begs the question, are longer reviews seen as being more helpful? To answer this question, let us examine the correlation between length and review_rating.

```
r_characters = cor.test(nchar(videogame$review),videogame$review_rating)
r_words = cor.test(str_count(string = videogame$review,pattern = '\\S+'),
                              videogame$review_rating)
r_sentences = cor.test(str_count(string = videogame$review,pattern = "[A-Za-z,;'\"\\s]+[^.!?]*[.?!]"),
                                  videogame$review_rating)

correlations = data.frame(r = c(r_characters$estimate, r_words$estimate, r_sentences$estimate),
                          p_value=c(r_characters$p.value, r_words$p.value, r_sentences$p.value))

rownames(correlations) = c('Characters','Words','Sentences')
correlations
```

	r <dbl>	p_value <dbl>
Characters	-0.08478651	1.016637e-43
Words	-0.08574868	1.111259e-44
Sentences	-0.09343258	9.478057e-53

3 rows

Grammar

Meaning may not only come from words but also grammatical elements such as punctuation and case. One may react differently to text that is in upper case compared to sentence case. Use of upper case may be seen as an expression of anger. Similarly, a set of consecutive exclamation marks may convey surprise or emphasis. Let us examine the prevalence of upper case letters and exclamation marks in the reviews.

Screaming Reviews

How do you feel about people who scream in texts or email? What about screaming in reviews?
Here we examine the proportion of upper case letters in a review.

```
percentUpper = 100*str_count(videogame$review,pattern='[A-Z]')/nchar(videogame$review)
summary(percentUpper)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.689   2.520   3.365   3.622 100.000
```

Exclamation Marks

In messaging, reviews and chat, we are allowed to break all rules of grammar like using a flood of exclamation marks. Does that tell us something?
First, how common are exclamations?

```
percentExclamation = 100*str_count(videogame$review,pattern='!')/nchar(videogame$review)
summary(percentExclamation)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   0.194   0.000  72.727
```

Are reviews with exclamations rated as being more helpful?

```
r_upper = cor.test(percentUpper,videogame$review_rating)
r_exclamation = cor.test(percentExclamation,videogame$review_rating)

correlations2 = data.frame(r = c(r_upper$estimate, r_exclamation$estimate),
                           p_value=c(r_upper$p.value, r_exclamation$p.value))

rownames(correlations2) = c('Upper Case','Exclamation Marks')
correlations2
```

	r <dbl>	p_value <dbl>
Upper Case	0.01865663	2.319852e-03
Exclamation Marks	0.05558568	1.073826e-19

2 rows

Keywords

Beyond meta-data on reviews, one may be interested in specific keywords in reviews. If we are curious to see what % of reviews mention 'minecraft' in them. We will use str_detect to look for Minecraft as a single word, two words, and independent of case. As you look at the results, bear in mind, these reviews are from 2013-14, only a few years after its release.

```
library(stringr)
mean(str_detect(string=tolower(videogame$review),pattern='minecraft|mine craft'))*100
```

```
## [1] 0.1913552
```

Console ~/

```
> str_detect(string=tolower(videogame$review),pattern='minecraft| mine craft')
   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [14] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [27] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [40] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
> sum(str_detect(string=tolower(videogame$review),pattern='minecraft| mine craft'))
[1] 51
>
> (sum(str_detect(string=tolower(videogame$review),pattern='minecraft| mine craft'))/nrow(videogame))*100
[1] 0.1913552
```

Tokenize

One approach to analyzing text is to treat it as a "bag of words" where words are examined independent of their position in text. To implement this approach, text is decomposed into tokens (tidytext::unnest_tokens). The default token is words, but it is possible to specify other tokens such as ngram, characters, and sentences. The words can be summarized and organized using dplyr functions. Later on, we will also use the tidyr package to reshape the data.

- The tidy text format as being a table with one-token-per-row

Words per Reviews

Let us begin by counting the number of words in each review.

```
library(dplyr); library(tidytext)
videogame%>%
  select(id,review)%>%
  unnest_tokens(output = word,input=review)%>%
  group_by(id)%>%
  summarize(count = n())%>%
  ungroup()
```

id <int>	count <int>
1	54
2	80
3	37
4	20
5	37
6	949
7	47
8	8
9	19
10	21

1-10 of 10,000 rows Previous 1 2 3 4 ... 1000 Next

- To select columns of a data frame, use select()
- To choose rows, use filter()

- Default tokenization in unnest_tokens() :
  - strips punctuation
  - converts characters to lowercase
  - is for single words

- The name word for the output column from unnest_tokens().
- The sentiment lexicons and stop word datasets have columns named word; performing inner joins and anti-joins is thus easier.

```
> videogame %>%
+   select(id,review)%>%
+   unnest_tokens(output = word,input=review)
      id                word
1      1                 1st
1.1    1            shipment
1.2    1            received
1.3    1                   a
1.4    1                book
1.5    1             instead
1.6    1                  of
1.7    1                 the
1.8    1                game
1.9    1                 2nd
1.10   1            shipment
1.11   1                 got
1.12   1                   a
1.13   1                fake
1.14   1                 one
1.15   1                game
1.16   1             arrived
1.17   1                with
1.18   1                   a
1.19   1               wrong
1.20   1                 key
1.21   1              inside
1.22   1                  on
1.23   1              sealed
1.24   1                 box
1.25   1                   i
1.26   1                 got
1.27   1                  in
1.28   1             contact
1.29   1                with
```

```
1.30   1           codemasters
1.31   1                   and
1.32   1                  send
1.33   1                  them
1.34   1              pictures
1.35   1                    of
1.36   1                   the
1.37   1                   dvd
1.38   1                   and
1.39   1                   the
1.40   1               content
1.41   1                  they
1.42   1                  said
1.43   1               nothing
1.44   1                  they
1.45   1                   can
1.46   1                    do
1.47   1                   its
1.48   1                     a
1.49   1                  fake
1.50   1          dvd.returned
1.51   1                    it
1.52   1                  good
1.53   1                   bye
2      2                     i
2.1    2                 still
2.2    2               haven't
2.3    2              figured
2.4    2                  this
2.5    2                   one
2.6    2                   out
2.7    2                   did
2.8    2             everything
2.9    2                    as
```

Filter

	id	review	review_rating
1	1	1st shipment received a book instead of the game.2nd ship...	1
2	2	I still haven't figured this one out. Did everything as instruct...	2
3	3	I bought this and the key didn't work. It was a gift, and the ...	1
4	4	I love it! Use it all the time. Really works perfectly with all ga...	5
5	5	my grandkids (and their parents) love playing with the micr...	5
6	6	The greatest bait and switch betrayal of a fanbase in videog...	1

```
2.10   2           instructed
2.11   2                  but
2.12   2                  the
2.13   2                 game
2.14   2                never
2.15   2            installed
2.16   2              strange
2.17   2                since
2.18   2                    i
2.19   2                don't
2.20   2                 like
2.21   2                   to
2.22   2                 rate
2.23   2                   on
2.24   2            something
2.25   2                    i
2.26   2              couldn't
2.27   2                  get
2.28   2                   to
2.29   2                 work
2.30   2                 i'll
2.31   2                 just
2.32   2                  say
2.33   2                 that
2.34   2                    i
2.35   2               didn't
2.36   2                 like
2.37   2                   it
2.38   2              because
2.39   2                    i
2.40   2              couldn't
2.41   2                  get
2.42   2                   it
2.43   2                   to
2.44   2                 work
2.45   2                    p
2.46   2                  for
2.47   2                those
2.48   2                 that
2.49   2                  did
2.50   2               manage
2.51   2                   to
2.52   2                  get
```

```
2.53   2                   it
2.54   2            installed
2.55   2                 i'm
2.56   2                sure
2.57   2                it's
2.58   2                   a
2.59   2                great
2.60   2                game
2.61   2                   i
2.62   2                 had
2.63   2                 the
2.64   2               first
2.65   2                dirt
2.66   2                 and
2.67   2                  it
2.68   2                 was
2.69   2                   a
2.70   2              scream
2.71   2                 too
2.72   2                 bad
2.73   2                this
2.74   2                 one
2.75   2              bombed
2.76   2               maybe
2.77   2                   a
2.78   2                 bad
2.79   2                disc
3      3                   i
3.1    3              bought
3.2    3                this
3.3    3                 and
3.4    3                 the
3.5    3                 key
3.6    3              didn't
3.7    3                work
3.8    3                  it
3.9    3                 was
```

Total words

So, what is the total number of words in all reviews?

```
videogame%>%
  select(id,review)%>%
  unnest_tokens(output = word,input=review)%>%
  count()  # count is a shortcut to summarize used above
```

n
<int>
3326753

1 row

Most Common Words

Let us see which words are used most frequently in these reviews. To do this, we will employ the tidytext library which uses a tidy data approach. Here are the top 25 words.

```
library(tidytext)
videogame%>%
  unnest_tokens(input = review, output = word)%>%
  select(word)%>%
  group_by(word)%>%
  summarize(count = n())%>%
  ungroup()%>%
  arrange(desc(count))%>%
  top_n(25)
```

word <chr>	count <int>
the	176348
and	93493
to	87413
a	84119
i	69741
of	62006
it	59193
is	59071
game	53600
you	51667

1-10 of 2… Previous **1** 2 Next

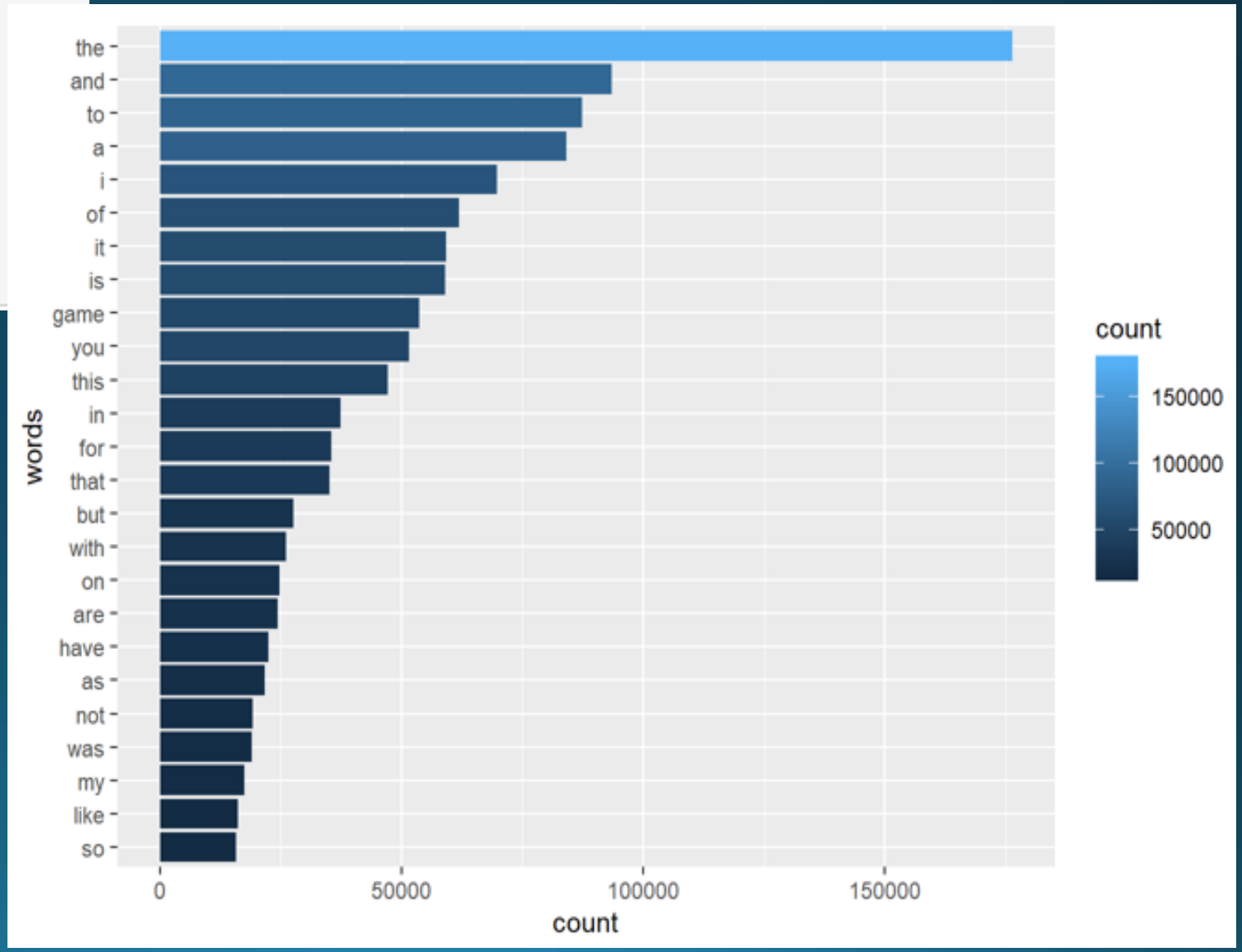
word <chr>	count <int>
this	47105
in	37400
for	35565
that	35185
but	27724
with	26205
on	24859
are	24292
have	22509
as	21664

11-20 of… Previous 1 … 3 Next

word <chr>	count <int>
not	19270
was	18995
my	17498
like	16219
so	15846

21-25 of… Previous 1 … **3** Next

```
videogame%>%
  unnest_tokens(input = review, output = word)%>%
  select(word)%>%
  group_by(word)%>%
  summarize(count = n())%>%
  ungroup()%>%
  arrange(desc(count))%>%
  top_n(25)%>%
  ggplot(aes(x=reorder(word,count), y=count, fill=count))+
    geom_col()+
    xlab('words')+
    coord_flip()
```



Not surprisingly the list contains a lot of prepositions and articles and words that don't convey much meaning. Such words are called stopwords. Let us look at the top 25 list after removing the stopwords.

In the code below, this is accomplished through an anti-join with a list of stop words, tidytext::stop_words.

```
videogame%>%
  unnest_tokens(input = review, output = word)%>%
  select(word)%>%
  anti_join(stop_words)%>%
  group_by(word)%>%
  summarize(count = n())%>%
  ungroup()%>%
  arrange(desc(count))%>%
  top_n(25)
```

- Excluding stop words.

COLUMBIA UNIVERSITY
School of Professional Studies

word <chr>	count <int>
game	53600
games	15024
play	11679
time	8582
fun	8539
34	7769
story	6268
playing	5538
graphics	5029
played	4878

1-10 of 2… Previous 1 2 Next

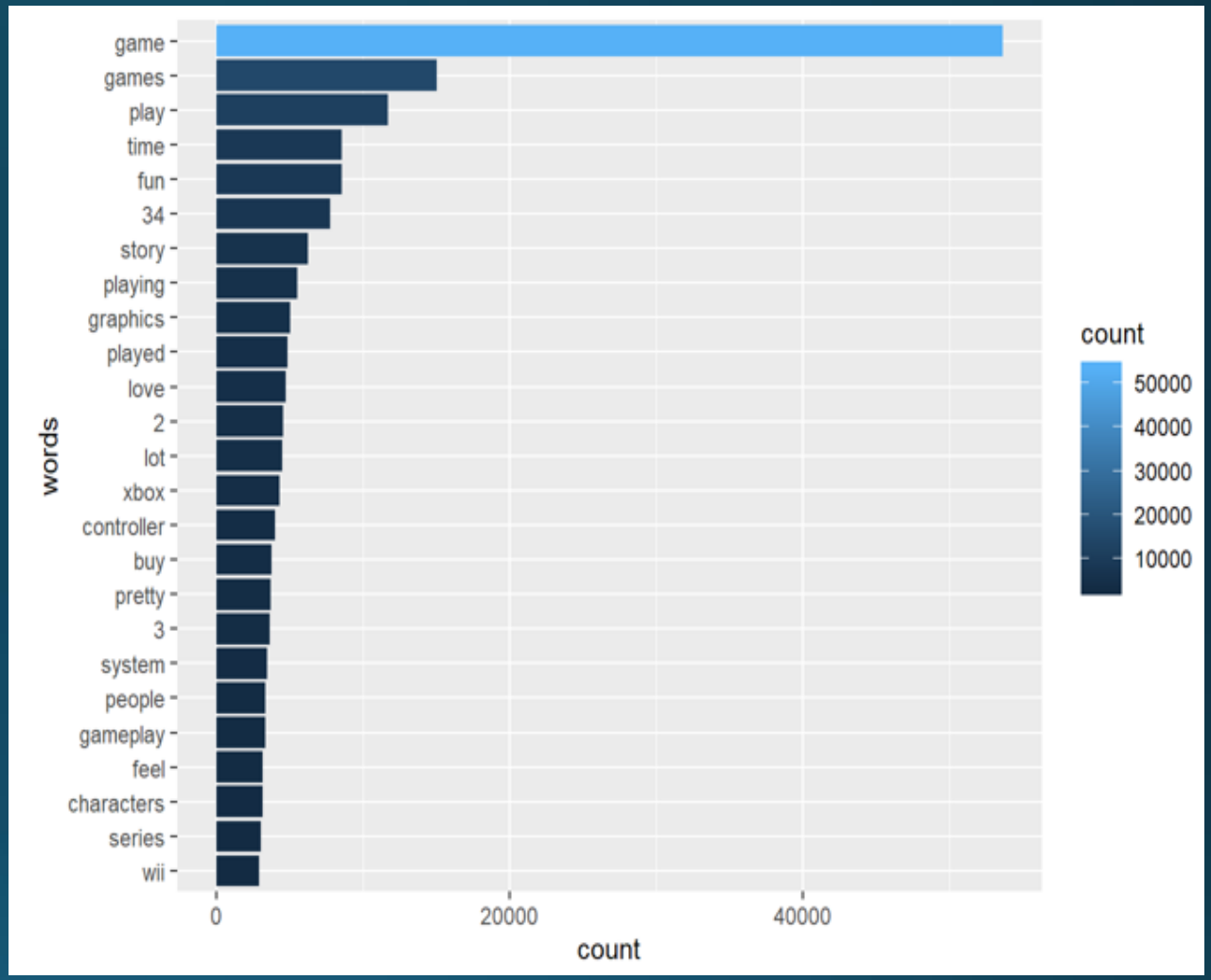
word <chr>	count <int>
love	4723
2	4562
lot	4478
xbox	4300
controller	4000
buy	3794
pretty	3736
3	3667
system	3462
people	3348

11-20 of… Previous 1 … 3 Next

word <chr>	count <int>
gameplay	3337
feel	3151
characters	3144
series	3015
wii	2933

21-25 of… Previous 1 … 3 Next

```r
videogame%>%
  unnest_tokens(input = review, output = word)%>%
  select(word)%>%
  anti_join(stop_words)%>%
  group_by(word)%>%
  summarize(count = n())%>%
  ungroup()%>%
  arrange(desc(count))%>%
  top_n(25)%>%
ggplot(aes(x=reorder(word,count), y=count, fill=count))+
    geom_col()+
    xlab('words')+
    coord_flip()
```



Now, set of top 25 words can also be generated using freq_terms from library(qdap), however we are not going to use this as installing qdap requires updating Java which has in the past posed a problem for some Mac users. In case you are interested, here is the code to generate the top 25 words.

```r
library(qdap) freq_terms(text.var = videogame$review,top = 25) plot(freq_terms(text.var = videogame$review,top = 25))
```

And, here is a list of top 25 words after removing stopwords based on the Top200Words list (rather than tidytext::stop_words).

```r
freq_terms(text.var=videogame$review,top=25,stopwords = Top200Words)  plot(freq_terms(text.var=videogame$review,top=25,stopwords = Top200Words))
```

Categorize

One of the simplest approaches to natural language processing is to categorize words based on their meaning. Words may be categorized based on their valence (positive or negative), emotion (e.g., happy, sad), domain (e.g., finance), among other things. This can be done conveniently using a relevant lexicon. There are a wide variety of lexicons that can be used based on the goal of the analysis. We will the process of categorizing tokens by using a set of lexicons.

Binary Sentiment (positive/negative) Lexicons

These lexicons use a simple approach of classifying tokens into two categories, usually positive or negative. One common lexicon for doing so is 'bing'

Bing Lexicon

The “bing” lexicon categorizes words as being positive or negative. The lexicon is included with the tidytext library and can be accessed by calling get_sentiments(‘bing’). Here are the first fifty words.

```
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

- tidytext provides a function get_sentiments() to get specific sentiment lexicons

	word <chr>	sentiment <chr>
11	abound	positive
12	abounds	positive
13	abrade	negative
14	abrasive	negative
15	abrupt	negative
16	abruptly	negative
17	abscond	negative
18	absence	negative
19	absent-minded	negative
20	absentee	negative

11-20 of 50 rows Previous 1 2 3 4 5 Next

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

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

2 rows

Valence of Words

Now, let us explore valence of the words used in reviews using the bing dictionary. We will match the words in the dictionary with the ones in the reviews to determine valence.

```
videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(get_sentiments('bing'))
```

```
> ?inner_join
>
```

Files | Plots | Packages | Help | Viewer

R: Mutating joins | Find in Topic

mutate-joins {dplyr}

Mutating joins

Description

The mutating joins add columns from `y` to `x`, matching rows based on the keys:

- `inner_join()`: includes all rows in `x` and `y`.

id <int>	review_rating <int>	word <chr>	sentiment <chr>
1	1	fake	negative
1	1	wrong	negative
1	1	fake	negative
1	1	good	positive
2	2	strange	negative
2	2	like	positive
2	2	work	positive
2	2	like	positive
2	2	work	positive
2	2	great	positive

1-10 of 10,000 rows Previous 1 2 3 4 … 1000 Next

id <int>	review_rating <int>	word <chr>	sentiment <chr>
2	2	dirt	negative
2	2	scream	negative
2	2	bad	negative
2	2	bad	negative
3	1	work	positive
3	1	problem	negative
3	1	good	positive
3	1	failed	negative
4	5	love	positive
4	5	works	positive

11-20 of 10,000 rows Previous 1 2 3 4 … 1000 Next

Valence in Reviews

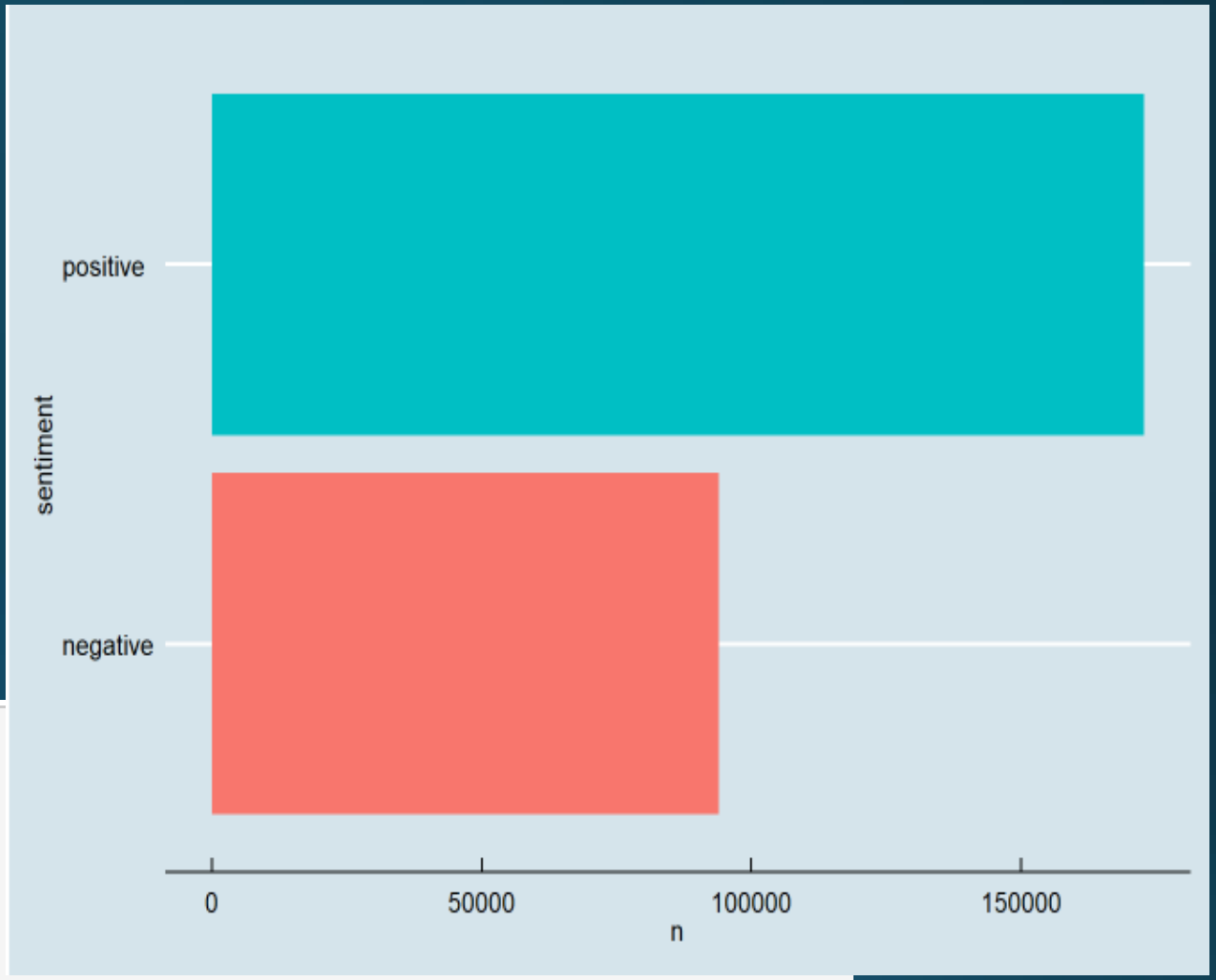
Let's find out the total number of positive and negative words in the reviews.

```
videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(get_sentiments('bing'))%>%
  group_by(sentiment)%>%
  count()%>%
  ungroup()
```

sentiment	n
<chr>	<int>
negative	93996
positive	172806

2 rows

Among all the review words, 93996 is categorized as negative




```
library(ggplot2)
library(ggthemes)

videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(get_sentiments('bing'))%>%
  group_by(sentiment)%>%
  count()%>%
  ungroup()%>%
  ggplot(aes(x=sentiment,y=n,fill=sentiment))+geom_col()+theme_economist()+guides(fill=F)+
  coord_flip()
```

Postitive Words

Proportion of Positive words

Next, let us find out the proportion of words in reviews that are positive. This is the ratio of number of positive words to sum of positive and negative words.

```
videogame %>%
  select(id,review)%>%
  unnest_tokens(output=word,input=review)%>%
  inner_join(get_sentiments('bing'))%>%
  group_by(sentiment)%>%
  summarize(n = n())%>%
  mutate(proportion = n/sum(n))%>%
  ungroup()
```

sentiment	n	proportion
<chr>	<int>	<dbl>
negative	93996	0.3523062
positive	172806	0.6476938

2 rows

Positive Words, helpful?

Let us drill down a bit more to see whether the proportion of positive words has any impact on its helpfulness. We will look at the proportion of positive (and negative words) for each rating.

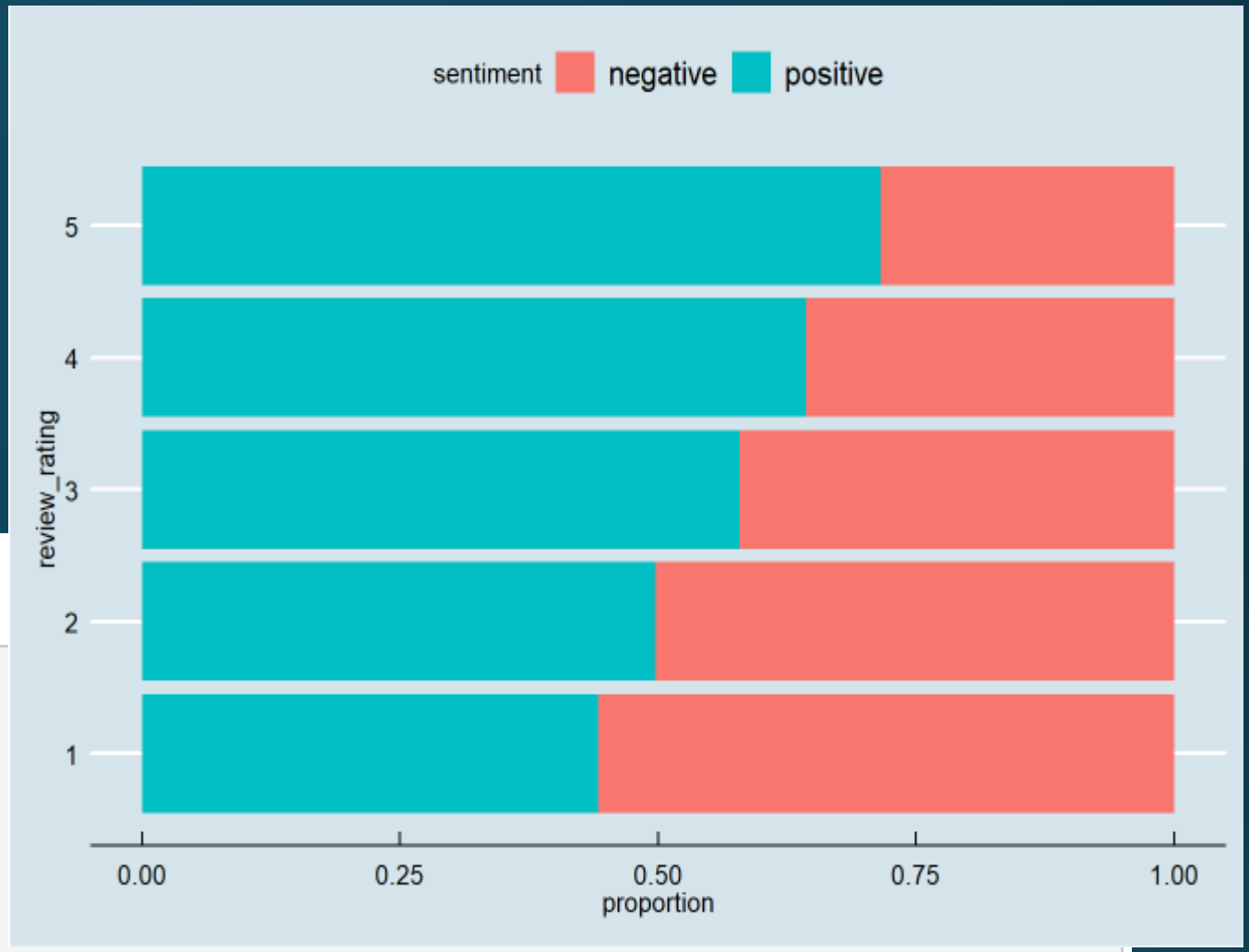
```
videogame %>%
  select(id,review,review_rating)%>%
  unnest_tokens(output=word,input=review)%>%
  inner_join(get_sentiments('bing'))%>%
  group_by(review_rating,sentiment)%>%
  summarize(n = n())%>%
  mutate(proportion = n/sum(n))%>%
  ungroup()
```

review_rating <int>	sentiment <chr>	n <int>	proportion <dbl>
1	negative	9104	0.5578431
1	positive	7216	0.4421569
2	negative	9051	0.5029451
2	positive	8945	0.4970549
3	negative	15186	0.4214821
3	positive	20844	0.5785179
4	negative	23600	0.3567703
4	positive	42549	0.6432297
5	negative	37055	0.2843669
5	positive	93252	0.7156331

- 5 possible review ratings
- Proportion of positive and negative words for each review rating
- Rating 1 has the highest proportion of negative words

and in pictures.

```
library(ggthemes)
videogame %>%
  select(id,review,review_rating)%>%
  unnest_tokens(output=word,input=review)%>%
  inner_join(get_sentiments('bing'))%>%
  group_by(review_rating,sentiment)%>%
  summarize(n = n())%>%
  mutate(proportion = n/sum(n))%>%
  ungroup()%>%
  ggplot(aes(x=review_rating,y=proportion,fill=sentiment))+geom_col()+theme_economist()+coord_flip()
```



Positive Reviews

Proportion of Positive Words per Review

Let us compute the proportion of positive words for each review. The proportion of positive words is the ratio of positive words and the sum of positive and negative words. This differs from the analysis above as it is computed for each review.

```
videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(get_sentiments('bing'))%>%
  group_by(id)%>%
  summarize(positive_words = sum(sentiment=='positive'),
            negative_words = sum(sentiment=='negative'),
            proportion_positive = positive_words/(positive_words+negative_words))%>%
  ungroup()
```

id <int>	positive_words <int>	negative_words <int>	proportion_positive <dbl>
1	1	3	0.25000000
2	5	5	0.50000000
3	2	2	0.50000000
4	3	0	1.00000000
5	3	0	1.00000000
6	31	52	0.37349398
7	1	3	0.25000000
8	2	1	0.66666667
9	3	0	1.00000000
10	2	0	1.00000000

1-10 of 10,000 rows Previous 1 2 3 4 5 6 ... 1000 Next

Correlation between Positive Words and Review Rating

Let us see if reviews with a lot of positive words are rated favorably.

```
videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(get_sentiments('bing'))%>%
  group_by(id,review_rating)%>%
  summarize(positive_words = sum(sentiment=='positive'),
            negative_words = sum(sentiment=='negative'),
            proportion_positive = positive_words/(positive_words+negative_words))%>%
  ungroup()%>%
  summarize(correlation = cor(proportion_positive,review_rating))
```

correlation
<dbl>
0.4023517

1 row

To examine the Correlation between Positive Words and Review Rating , we need to first calculate fraction of positive words in each review.

NRC Sentiment Polarity Lexicon

There are a couple of nrc lexicons. This one is a binary sentiment lexicon that categorizes words as +1 or -1

```
library(lexicon)
head(hash_sentiment_nrc)
```

x	y
<chr>	<int>
abandon	-1
abandoned	-1
abandonment	-1
abba	1
abduction	-1
aberrant	-1

6 rows

```
hash_sentiment_nrc %>%
  group_by(y)%>%
  summarize(count= n())%>%
  ungroup()
```

y	count
<int>	<int>
-1	3241
1	2227

2 rows

Since lexicons vary by number of words covered and manner of classification, results will vary by lexicon.

```
videogame %>%
  select(id, review)%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(y = hash_sentiment_nrc,by = c('word'='x'))%>%
  group_by(y)%>%
  summarize(count = n())%>%
  ungroup()
```

y <int>	count <int>
-1	88608
1	152018

2 rows

Emotion Lexicon

This family of lexicons categorizes words based on the emotion conveyed. Rather than grouping words as positive or negative, words are categorized based on the emotion reflected. Furthermore, a single word may reflect multiple emotions.

NRC Emotion Lexicon

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 and later in 2019 re-integrated. To access `nrc` from `tidytext`, run `get_sentiments('nrc')` in interactive mode (i.e., console or script, not markdown) and agree to the non-commercial use agreement. This should only be required for the first-use. Another alternative is to use the lexicon that I have 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.

	word	sentiment
1	abacus	trust
2	abandon	fear
3	abandon	negative
4	abandon	sadness
5	abandoned	anger
6	abandoned	fear
7	abandoned	negative
8	abandoned	sadness
9	abandonment	anger
10	abandonment	fear
11	abandonment	negative
12	abandonment	sadness
13	abandonment	surprise
14	abba	positive
15	abbot	trust
16	abduction	fear
17	abduction	negative
18	abduction	sadness
19	abduction	surprise
20	aberrant	negative

Method 1: Get Lexicon from `tidytext`

```
nrc = get_sentiments('nrc')
```

Method 2: Get Lexicon posted on github

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

	word	sentiment	num
1	aback	anger	0
2	aback	anticipation	0
3	aback	disgust	0
4	aback	fear	0
5	aback	joy	0
6	aback	negative	0
7	aback	positive	0
8	aback	sadness	0
9	aback	surprise	0
10	aback	trust	0
11	abacus	anger	0
12	abacus	anticipation	0
13	abacus	disgust	0
14	abacus	fear	0
15	abacus	joy	0
16	abacus	negative	0
17	abacus	positive	0
18	abacus	sadness	0
19	abacus	surprise	0
20	abacus	trust	1
21	abandon	anger	0
22	abandon	anticipation	0
23	abandon	disgust	0
24	abandon	fear	1
25	abandon	joy	0
26	abandon	negative	1
27	abandon	positive	0
28	abandon	sadness	1
29	abandon	surprise	0
30	abandon	trust	0

	word	sentiment
20	abacus	trust
24	abandon	fear
26	abandon	negative
28	abandon	sadness
31	abandoned	anger
34	abandoned	fear
36	abandoned	negative
38	abandoned	sadness
41	abandonment	anger
44	abandonment	fear
46	abandonment	negative
48	abandonment	sadness
49	abandonment	surprise
77	abba	positive
90	abbot	trust
134	abduction	fear
136	abduction	negative
138	abduction	sadness
139	abduction	surprise
146	aberrant	negative

Here is a list of emotions covered by this lexicon.

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

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

```
table(nrc$sentiment)
```

```
##
##       anger anticipation      disgust         fear          joy     negative
##        1247          839         1058         1476          689         3324
##    positive      sadness     surprise        trust
##        2312         1191          534         1231
```

Emotions in Reviews

Let us examine the emotions expressed in the reviews

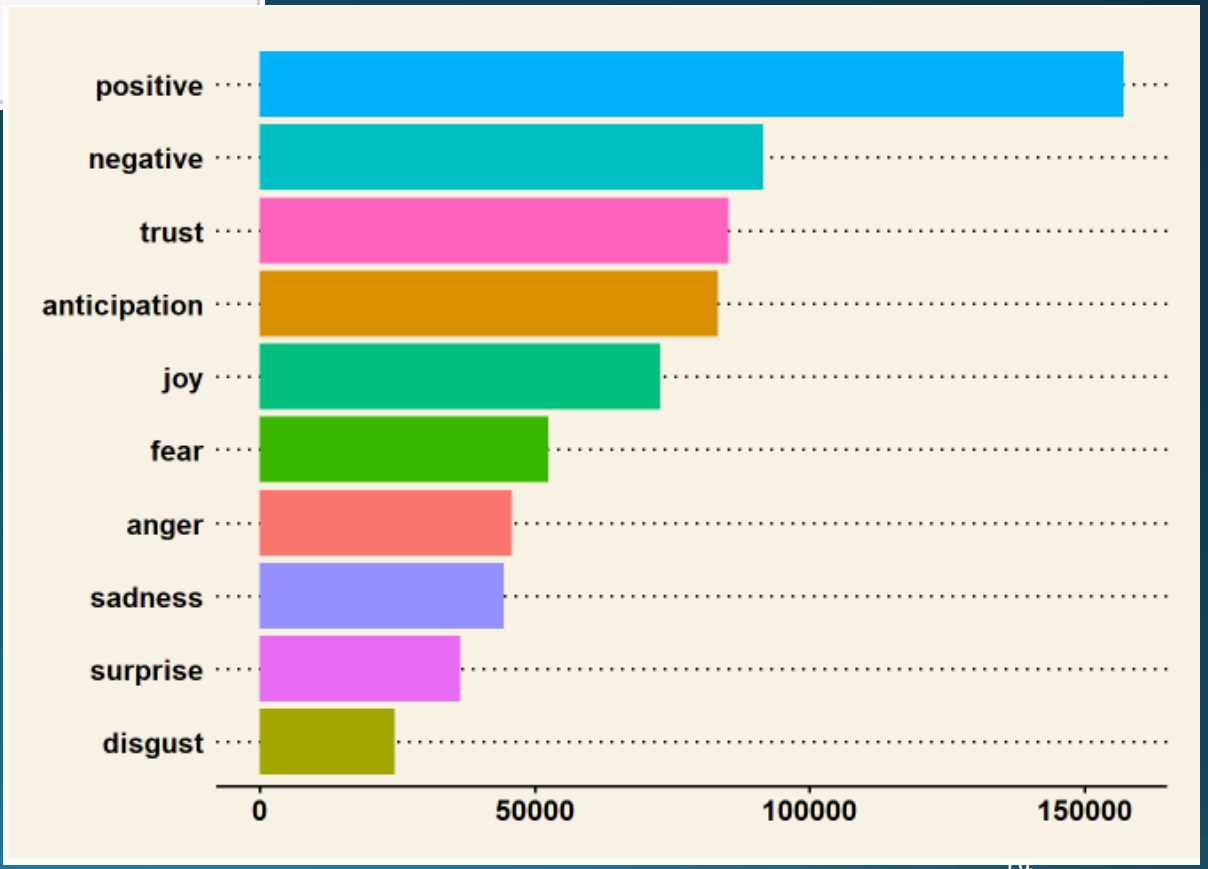
```
videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(nrc)%>%
  group_by(sentiment)%>%
  count()%>%
  arrange(desc(n))%>%
  ungroup()
```

sentiment <chr>	n <int>
positive	157200
negative	91605
trust	85287
anticipation	83399
joy	72826
fear	52559
anger	45833
sadness	44438
surprise	36535
disgust	24478

1-10 of 10 rows

- 72826 words reflect joy

```
videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(nrc)%>%
  group_by(sentiment)%>%
  count()%>%
  ungroup()%>%
  ggplot(aes(x=reorder(sentiment,X = n),y=n,fill=sentiment))+
        geom_col()+guides(fill=F)+coord_flip()+theme_wsj()
```



id <int>	sentiment <chr>	n <int>
1	anticipation	2
1	joy	2
1	negative	3
1	positive	4
1	surprise	1
1	trust	2
2	anger	3
2	disgust	5
2	fear	3
2	negative	5

1-10... Previous 1 2 Next

id <int>	sentiment <chr>	n <int>
2	positive	1
2	sadness	2
2	surprise	1
2	trust	1
3	anticipation	3
3	fear	1
3	joy	3
3	negative	1
3	positive	3
3	sadness	1

11-... Previous 1 ... 3 Next

id <int>	sentiment <chr>	n <int>
3	surprise	2
3	trust	2
4	anticipation	1
4	joy	1
4	positive	1
5	anticipation	1
5	joy	3
5	positive	5
5	sadness	1
5	trust	1

21-... Previous 1 ... 4 Next

Emotion in Each Review

Here, we will examine each review and the frequency of each emotion expressed.

```
library(tidyr)
videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(nrc)%>%
  group_by(id,sentiment)%>%
  count()%>%
  pivot_wider(names_from = sentiment,values_from=n)%>%
  select(id, positive, negative, trust, anticipation,
         joy, fear, anger, sadness, surprise, disgust)%>%
  mutate_at(.vars = 2:11, .funs = function(x) replace_na(x,0))%>%
  ungroup()
```

id <int>	positive <dbl>	negative <dbl>	trust <dbl>	anticipation <dbl>	joy <dbl>	fear <dbl>	anger <dbl>	sadness <dbl>	surprise <dbl>	disgust <dbl>
1	4	3	2	2	2	0	0	0	1	0
2	1	5	1	0	0	3	3	2	1	5
3	3	1	2	3	3	1	0	1	2	0
4	1	0	0	1	1	0	0	0	0	0
5	5	0	1	1	3	0	0	1	0	0
6	54	46	35	14	19	21	22	24	8	28
7	1	3	1	0	0	1	1	0	1	0
8	0	1	0	0	0	0	0	0	0	0
9	1	0	1	1	1	0	0	0	1	0
10	0	0	1	0	0	0	0	0	0	0

Emotions and Rating

One may suspect that review rating will be tied to the emotion expressed. Let us explore this possibility.

```
videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(nrc)%>%
  group_by(id,sentiment,review_rating)%>%
  count()%>%
  pivot_wider(names_from = sentiment,values_from = n)%>%
  mutate_at(.vars = 3:12, .funs = function(x) replace_na(x,0))%>%
  pivot_longer(cols = 3:12, names_to = 'sentiment',values_to = 'n')%>%
  ungroup()%>%
  group_by(sentiment, review_rating)%>%
  summarize(n = mean(n))%>%
  ungroup()
```

id <int>	sentiment <chr>	review_rating <int>	n <int>
1	anticipation	1	2
1	joy	1	2
1	negative	1	3
1	positive	1	4
1	surprise	1	1
1	trust	1	2
2	anger	2	3
2	disgust	2	5
2	fear	2	3
2	negative	2	5

1-10 of 162,... Previous 1 2 3 4 ... 100 Next

id <int>	sentiment <chr>	review_rating <int>	n <int>
2	positive	2	1
2	sadness	2	2
2	surprise	2	1
2	trust	2	1
3	anticipation	1	3
3	fear	1	1
3	joy	1	3
3	negative	1	1
3	positive	1	3
3	sadness	1	1

11-20 of 16... Previous 1 2 3 4 ... 100 Next

id <int>	sentiment <chr>	review_rating <int>	n <int>
3	surprise	1	2
3	trust	1	2
4	anticipation	5	1
4	joy	5	1
4	positive	5	1
5	anticipation	5	1
5	joy	5	3
5	positive	5	5
5	sadness	5	1
5	trust	5	1

21-30 of 16... Previous 1 2 3 4 ... 100 Next

sentiment <chr>	review_rating <int>	n <dbl>
anger	1	2.4053030
anger	2	3.1192369
anger	3	2.4467466
anger	4	2.2811951
anger	5	1.3365469
anticipation	1	2.8705808
anticipation	2	4.0715421
anticipation	3	4.0192657
anticipation	4	4.2058192
anticipation	5	2.7927897

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

id <int>	review_rating <int>	sentiment <chr>	n <dbl>
1	1	anticipation	2
1	1	joy	2
1	1	negative	3
1	1	positive	4
1	1	surprise	1
1	1	trust	2
1	1	anger	0
1	1	disgust	0
1	1	fear	0
1	1	sadness	0

1-10 of 254,... Previous 1 2 3 4 ... 100 Next

id <int>	review_rating <int>	sentiment <chr>	n <dbl>
2	2	anticipation	0
2	2	joy	0
2	2	negative	5
2	2	positive	1
2	2	surprise	1
2	2	trust	1
2	2	anger	3
2	2	disgust	5
2	2	fear	3
2	2	sadness	2

11-20 of 25... Previous 1 2 3 4 ... 100 Next

```
videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(nrc)%>%
  group_by(id,sentiment,review_rating)%>%
  count()%>%
  pivot_wider(names_from = sentiment,values_from = n)%>%
  mutate_at(.vars = 3:12, .funs = function(x) replace_na(x,0))%>%
  pivot_longer(cols = 3:12, names_to = 'sentiment',values_to = 'n')%>%
  ungroup()%>%
  group_by(sentiment, review_rating)%>%
  summarize(n = mean(n)) %>%
  ungroup()%>%
  ggplot(aes(x=review_rating,y=n,fill=review_rating))+
  geom_col()+
  facet_wrap(~sentiment)+
  guides(fill=F)+
  coord_flip()+
  theme_bw()
```

Correlation between emotion expressed and review rating, by sentiment

Let us quantify the relationship by examining the correlation between frequency of emotions expressed and rating, by sentiment. Can you explain these results?

```
videogame%>%
  unnest_tokens(output = word, input = review)%>%
  inner_join(nrc)%>%
  group_by(id,sentiment,review_rating)%>%
  count()%>%
  pivot_wider(names_from = sentiment,values_from = n)%>%
  mutate_at(.vars = 3:12, .funs = function(x) replace_na(x,0))%>%
  pivot_longer(cols = 3:12, names_to = 'sentiment',values_to = 'n')%>%
  ungroup()%>%
  group_by(sentiment)%>%
  summarize(r = cor(n,review_rating))%>%
  ungroup()
```

sentiment <chr>	r <dbl>
anger	-0.11223672
anticipation	-0.05758537
disgust	-0.14776351
fear	-0.09941267
joy	-0.01102204
negative	-0.14328531
positive	-0.03517622
sadness	-0.11809721
surprise	-0.06038202
trust	-0.05684609

1-10 of 10 rows

Sentiment Score Lexicons

afinn Sentiment

The bing and nrc emotion lexicons classify a word based on the presence or absence of an emotion or valence. The afinn lexicon scores each word based on the extent to which it is positive or negative. For instance, the afinn lexicon will make a distinction between words “satisfied” and “delighted” based on how positive they are, but the bing lexicon will simply categorize both as being positive.

```
get_sentiments('bing')[get_sentiments('bing')$word %in% c('satisfied','delighted'), ]
```

word <chr>	sentiment <chr>
delighted	positive
satisfied	positive

2 rows

The afinn lexicon assigns a numeric value to the emotion. Thus this lexicon can be thought of as computing a sentiment score for words.

```
get_sentiments('afinn')[get_sentiments('afinn')$word %in% c('satisfied','delighted'),]
```

word <chr>	value <dbl>
delighted	3
satisfied	2

2 rows

This lexicon which was previously a part of library(tidytext) was dropped from the package as of June 14, 2019 and later in 2019 re-integrated. To access `afinn` from `tidytext`, run `get_sentiments('afinn')` in interactive mode (i.e., console or script, not markdown) and agree to the non-commercial use agreement. This should only be required for the first-use. Another alternative is to use the lexicon that I have 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.

Method 1: Get Lexicon from `tidytext`

```
afinn = get_sentiments('afinn')
```

Method 2: Get Lexicon from github

```
afinn = read.table('https://raw.githubusercontent.com/pseudorational/data/master/AFINN-111.txt',
                   header = F,
                   quote="",
                   sep = '\t',
                   col.names = c('word','value'),
                   encoding='UTF-8',
                   stringsAsFactors = F)
```

```
afinn[1:50,]
```

	word <chr>	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()%>%
  ungroup()
```

value <int>	n <int>
-5	16
-4	43
-3	264
-2	966
-1	309
0	1
1	208
2	448
3	172
4	45

1-10 of 11 rows Previous 1 2 Next

value <int>	n <int>
5	5

11-11 of 11 rows Previous 1 2 Next

Review 617

Now, let's use this dictionary to determine the sentiment of review 617.

```
videogame %>%
  select(id,review)%>%
  unnest_tokens(output=word,input=review)%>%
  inner_join(afinn)%>%
  filter(id==617)%>%
  summarize(reviewSentiment = mean(value))
```

reviewSentiment
<dbl>
2.066667

1 row

Here is the review

```
videogame$review[617]
```

```
## [1] "Like scottrocket3 said, you're out of your damn mind if you buy this for $200.  This is the best Mar
io 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 wrest
ling great Captain Lou Albano who, unfortunately passed away recently.  He was a Christian, so I will see hi
m 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 t
he truckload!!  They're good for you & have vitamin D, the sunshine vitamin."
```

What is the minimum, maximum, median, or average sentiment across all reviews?

All Reviews

```
videogame %>%
  select(id,review)%>%
  unnest_tokens(output=word,input=review)%>%
  inner_join(afinn)%>%
  group_by(id)%>%
  summarize(reviewSentiment = mean(value))%>%
  summarize(min=min(reviewSentiment),max=max(reviewSentiment),
            median=median(reviewSentiment),mean=mean(reviewSentiment))%>%
  ungroup()
```

min	max	median	mean
<dbl>	<dbl>	<dbl>	<dbl>
-4	5	1.444444	1.37611

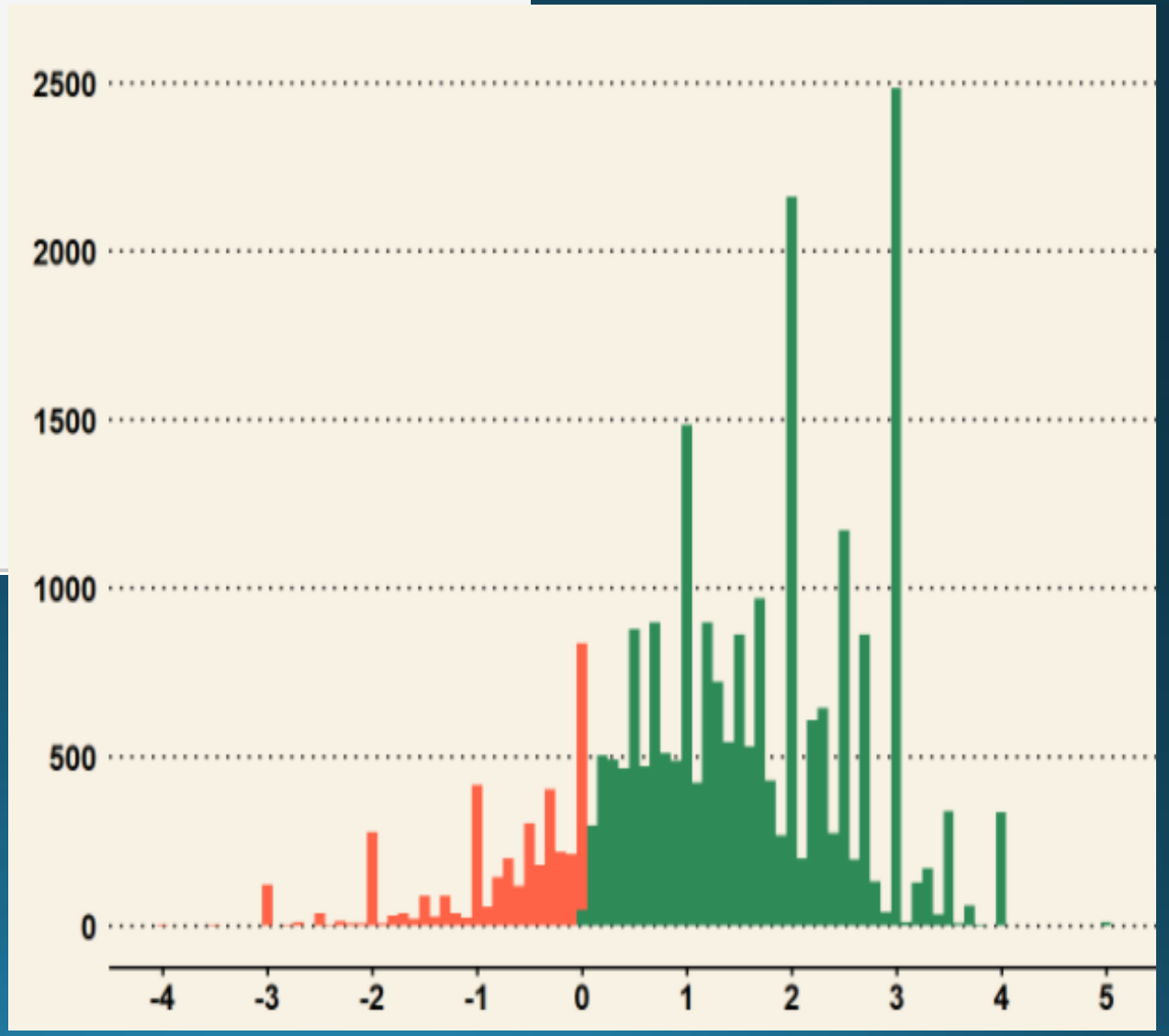
1 row

```
> videogame %>%
+     select(id,review)%>%
+     unnest_tokens(output=word,input=review)%>%
+     inner_join(afinn)%>%
+     group_by(id)%>%
+     summarize(reviewSentiment = mean(value))
Joining, by = "word"
# A tibble: 25,839 x 2
      id reviewSentiment
   <int>           <dbl>
 1     1          -1.25
 2     2          -0.5
 3     3           0.4
 4     4           3
 5     5           2.5
 6     6          -0.647
 7     7          -0.667
 8     8           0.5
 9     9           2.5
10    10           3
# ... with 25,829 more rows
```

```
> videogame %>%
+   select(id,review)%>%
+   unnest_tokens(output=word,input=review)%>%
+   inner_join(afinn)%>%
+   group_by(id)%>%
+   summarize(reviewSentiment = mean(value))%>%
+   filter(reviewSentiment==-4)
Joining, by = "word"
# A tibble: 3 x 2
     id reviewSentiment
  <int>           <dbl>
1  6587              -4
2  7583              -4
3 16654              -4
```

Reviews with id which have the lowest sentiment score

```
videogame %>%
  select(id,review)%>%
  unnest_tokens(output=word,input=review)%>%
  inner_join(afinn)%>%
  group_by(id)%>%
  summarize(reviewSentiment = mean(value))%>%
  ungroup()%>%
  ggplot(aes(x=reviewSentiment,fill=reviewSentiment>0))+
  geom_histogram(binwidth = 0.1)+
  scale_x_continuous(breaks=seq(-5,5,1))+
  scale_fill_manual(values=c('tomato','seagreen'))+
  guides(fill=F)+
  theme_wsj()
```



Jockers Lexicon

Here is another sentiment lexicon. Scores for words range from -1 to +1.

```
library(lexicon)
head(key_sentiment_jockers)
```

	word <chr>	value <dbl>
1	abandon	-0.75
2	abandoned	-0.50
3	abandoner	-0.25
4	abandonment	-0.25
5	abandons	-1.00
6	abducted	-1.00

6 rows

Let us examine Reviews sentiment based on this lexicon

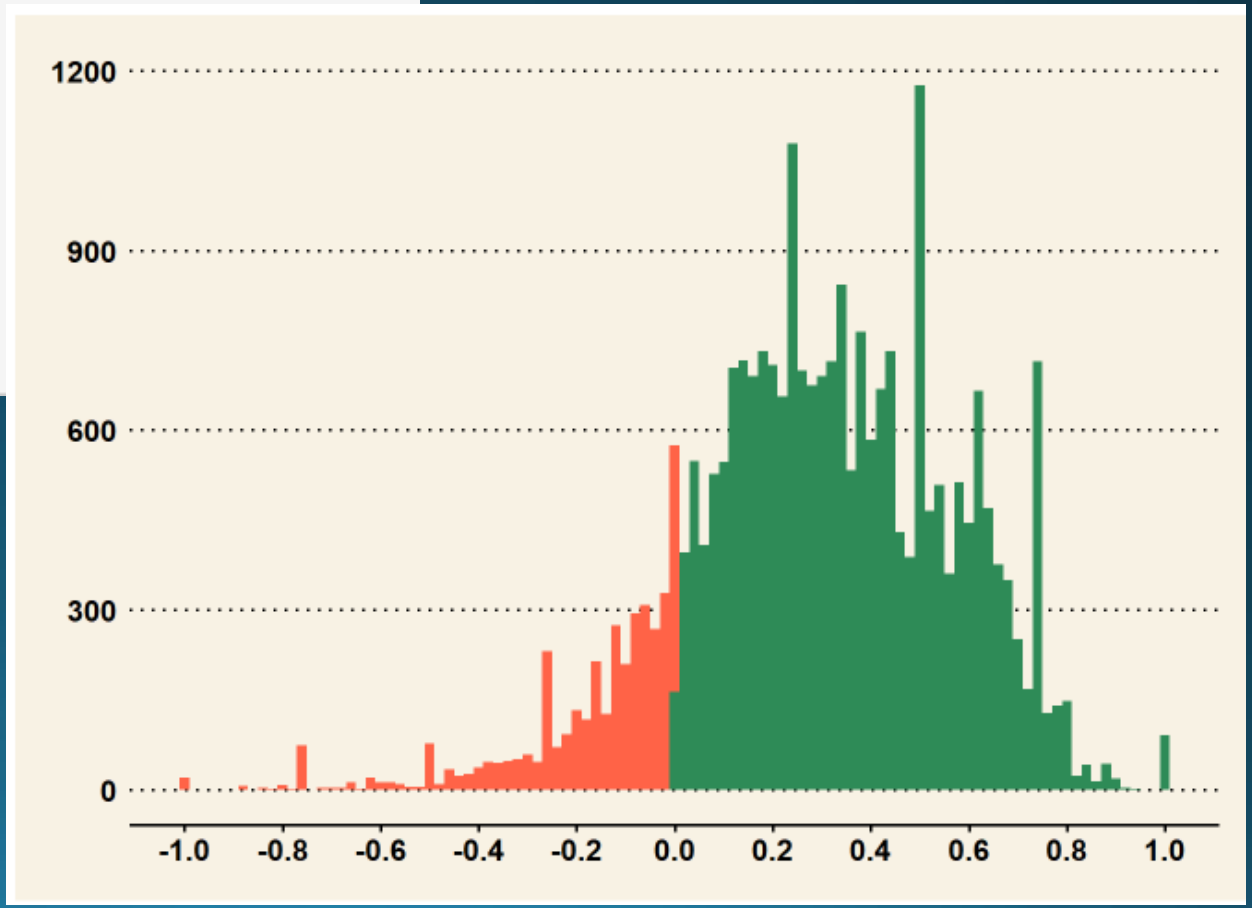
```
videogame %>%
  select(id,review)%>%
  unnest_tokens(output=word,input=review)%>%
  inner_join(key_sentiment_jockers)%>%
  group_by(id)%>%
  summarize(reviewSentiment = mean(value))%>%
  summarize(min=min(reviewSentiment),max=max(reviewSentiment),
            median=median(reviewSentiment),mean=mean(reviewSentiment))%>%
  ungroup()
```

min <dbl>	max <dbl>	median <dbl>	mean <dbl>
-1	1	0.2991525	0.2933024

1 row

Visualizing Sentiment scores


```
videogame %>%
  select(id,review)%>%
  group_by(id)%>%
  unnest_tokens(output=word,input=review)%>%
  inner_join(key_sentiment_jockers)%>%
  summarize(reviewSentiment = mean(value))%>%
  ungroup()%>%
  ggplot(aes(x=reviewSentiment,fill=reviewSentiment>0))+
  geom_histogram(binwidth = 0.02)+
  scale_x_continuous(breaks=seq(-1,1,0.2))+
  scale_fill_manual(values=c('tomato','seagreen'))+
  guides(fill=F)+
  theme_wsj()
```



Senticnet Lexicon

Here is yet another sentiment score lexicon with scores ranging from -1 to +1.

```
library(lexicon)
head(hash_sentiment_senticnet)
```

x <chr>	y <dbl>
aaa	0.606
aah	-0.510
abandon	-0.560
abandonment	-0.650
abase	-0.580
abasement	-0.580

6 rows

Let us examine Reviews sentiment based on this lexicon

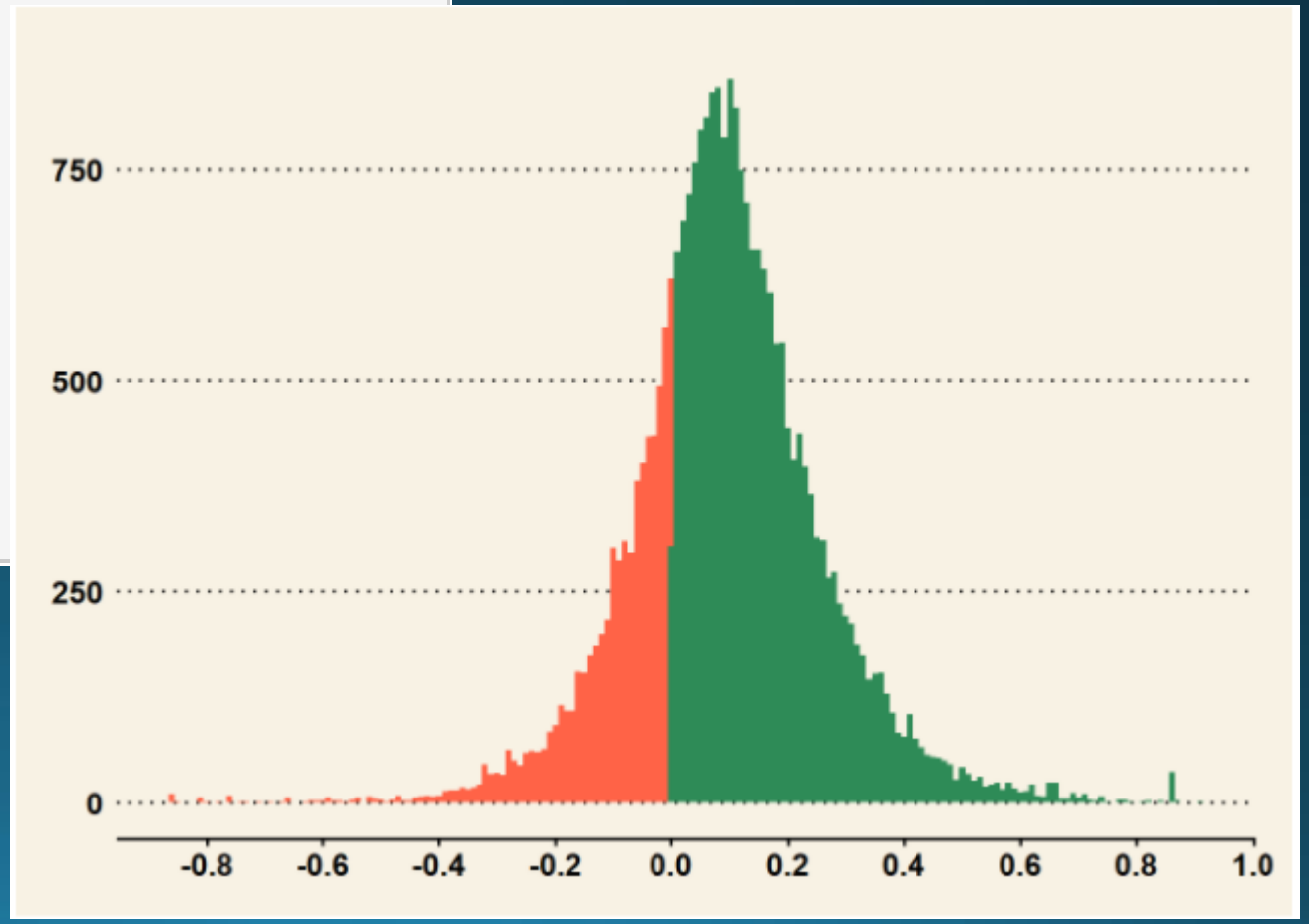
```
videogame %>%
  select(id,review)%>%
  unnest_tokens(output=word,input=review)%>%
  inner_join(hash_sentiment_senticnet, by = c('word'='x'))%>%
  group_by(id)%>%
  summarize(reviewSentiment = mean(y))%>%
  summarize(min=min(reviewSentiment),max=max(reviewSentiment),
            median=median(reviewSentiment),mean=mean(reviewSentiment)) %>%
  ungroup()
```

min <dbl>	max <dbl>	median <dbl>	mean <dbl>
-0.86	0.909	0.08878472	0.09478637

1 row

Visualizing Sentiment scores


```
videogame %>%
  select(id,review)%>%
  group_by(id)%>%
  unnest_tokens(output=word,input=review)%>%
  inner_join(hash_sentiment_senticnet, by = c('word'='x'))%>%
  summarize(reviewSentiment = mean(y))%>%
  ungroup()%>%
  ggplot(aes(x=reviewSentiment,fill=reviewSentiment>0))+
  geom_histogram(binwidth = 0.01)+
  scale_x_continuous(breaks=seq(-1,1,0.2))+
  scale_fill_manual(values=c('tomato','seagreen'))+
  guides(fill=F)+
  theme_wsj()
```



Other Lexicons

There are of course many other lexicons.

```
head(hash_sentiment_sentiword)
```

x <chr>	y <dbl>
365 days	-0.5000
366 days	0.2500
3tc	-0.2500
a fortiori	0.2500
a good deal	0.2500
a great deal	0.3125

6 rows

```
hash_sentiment_socal_google
```

x <chr>	y <dbl>
a pillar	2.904543116
ab liva	-0.957870011
able	2.639373993
above average	3.215001812
above mentioned	2.581580349
abrasive	1.675191287
absent	0.485064882
absolute	0.473787953
abstract	2.006855726
absurd	-3.777874410

1-10 of 3,290 rows Previous 1 2 ... 329 Next

```
hash_sentiment_slangsd
```

x <chr>	y <dbl>
a a	-0.5
a bad amount of money	-0.5
a bad mother	-0.5
a bad taste in my mouth	-0.5
a bag o' beagles	-0.5
a bag of old and moldy ass	-0.5
a barlow	-0.5
a barney	-0.5
a batman	0.5
a bitch	-0.5

1-10 of 10,000 rows Previous 1 2 3 4 ... 1000 Next

Profanity

Under certain scenarios, one may want to filter out words such as when the words don't add additional information or if they are offensive.

Here are a few lexicons that contain offensive words that one may want to use to filter out from one's dataset.

- profanity_alvarez
- profanity_arr_bad
- profanity_racist
- profanity_zac_anger

To illustrate, here we are removing offensive words using two lexicons from videogame reviews

```
videogame %>%
  unnest_tokens(output = word, input = review)%>%
  select(id, word)%>%
  anti_join(y = data.frame(word = profanity_racist),
            by=c('word'='word'))
```

Console | Terminal | Jobs

R 4.1.2 · ~/

```
> profanity_racist
  [1] "abeed"
  [2] "abid"
  [3] "africoon"
  [4] "alligator bait"
  [5] "arabush"
  [6] "arabushs"
  [7] "armo"
  [8] "armos"
  [9] "banana"
 [10] "bananas"
 [11] "beanbag"
 [12] "beanbags"
 [13] "beaner"
 [14] "beaners"
 [15] "beaney"
 [16] "beaneys"
 [17] "bluegum"
 [18] "bluegums"
 [19] "boche"
 [20] "boches"
 [21] "bong"
 [22] "boong"
 [23] "boonga"
```

id	word
<int>	<chr>
1	1st
1	shipment
1	received
1	a
1	book
1	instead
1	of
1	the
1	game
1	2nd

1-10 of 10,000 rows Previous **1** 2 3 4 … 1000 Next

Filtering joins

Description

Filtering joins filter rows from `x` based on the presence or absence of matches in `y`:

- `semi_join()` return all rows from `x` with a match in `y`.
- `anti_join()` return all rows from `x` without a match in `y`.

Wordcloud

In general, wordclouds offer little insight into the data, yet they tend to be very good at capturing interest of non-technical audiences. Let us begin by creating a wordcloud from our data using library(tidytext), library(dplyr), library(tidyr),and library(wordcloud) functions.

```
wordcloudData =
  videogame%>%
  unnest_tokens(output=word,input=review)%>%
  anti_join(stop_words)%>%
  group_by(word)%>%
  summarize(freq = n())%>%
  arrange(desc(freq))%>%
  ungroup()%>%
  data.frame()

library(wordcloud)
set.seed(617)
wordcloud(words = wordcloudData$word,wordcloudData$freq,scale=c(2,0.5),
          max.words = 100,colors=brewer.pal(9,"Spectral"))
```

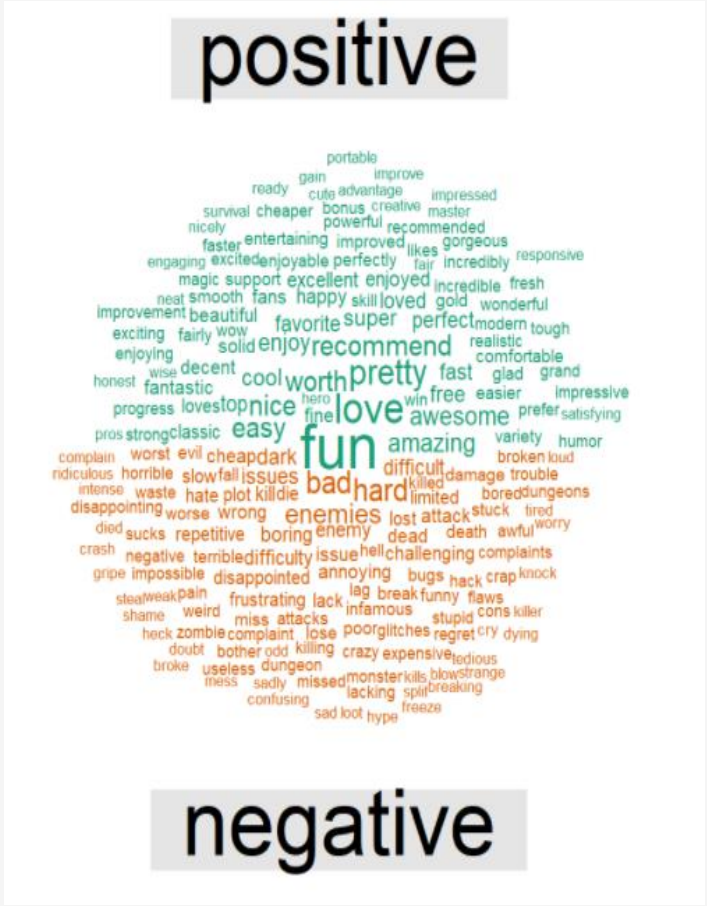
Comparison Cloud

Finally, here is a comparison cloud to contrast positive and negative words in the reviews.

```
library(tidyr)
wordcloudData =
  videogame%>%
  unnest_tokens(output=word,input=review)%>%
  anti_join(stop_words)%>%
  inner_join(get_sentiments('bing'))%>%
  count(sentiment,word,sort=T)%>%
  spread(key=sentiment,value = n,fill=0)%>%
  data.frame()

rownames(wordcloudData) = wordcloudData[,'word']
wordcloudData = wordcloudData[,c('positive','negative')]

set.seed(617)
comparison.cloud(term.matrix = wordcloudData,scale = c(2,0.5),max.words = 200, rot.per=0)
```



	positive	negative
abnormal	0	2
abominable	0	1
abomination	0	14
abort	0	2
aborted	0	5
abound	11	0
abounds	2	0
abrupt	0	6
abruptly	0	5
absence	0	15
absurd	0	31
absurdity	0	5
absurdly	0	14
abundance	12	0
abundant	5	0
abuse	0	30
abused	0	10
abuses	0	1
abusive	0	7
abysmal	0	21
abysmally	0	6
abyss	0	44
accessable	1	0
accessible	80	0
accidental	0	11
acclaim	6	0
acclaimed	16	0
accolades	4	0

Showing 1 to 30 of 3,987 entries, 2 total columns