Monty Python's Flying Circus (of Text Analysis in R!)

Locke Data

NHS-R

Welcome!

Today's aim

- Learn the fundamentals of text analysis in R
- Have fun doing it

Polls

Hands up if you ...

- know the difference between a `table` and a `data.frame` in R
- use `dplyr`
- have done webscraping in R
- are an NLP expert

Agenda

- The Meaning of Life
  - Some initial concepts, a foray into text analytics the "cheating way", and some fundamentals
- The Life of Brian
  - Learning how to perform simple text analysis tasks
- The Holy Grail
  - Performing more advanced text analysis

Let's get ready

- Login details
- `TextAnalysis` package
- tidytextmining.com

Act 1: The Meaning of Life

Why do text analysis?

- Trump's tweets
- Categorising documents
- Intervene before a suicide
- Fraud detection

Core text terms

- **word**
- **stop word** a commonly used word
- **token** a word or another meaningful grouping of text
- **n-gram** an n-long seqence of words
- **sentiment** the feeling indicated by the text
- **topic** what the text is talking about

- **lexicon** vocabulary / dictionary

How things hang together in R

- Get text data
  - Use `rvest` to scrape web sites
- Clean data
  - Use `dplyr` for manipulation
  - Use `tidytext` for text tokenisation & more
- Summarise data
  - Use `tidytext`
- Visualise data
  - Use `ggplot2`

Let's start simple!

Use someone else's hardwork e.g. Microsoft Cognitive Services

microsoft.com/cognitive-services

> [The service] returns a score between 0 and 1 denoting overall sentiment in the input text. Scores close to 1 indicate positive sentiment, while scores close to 0 indicate negative sentiment.
>
> Sentiment score is generated using classification techniques. The input features to the classifier include n-grams, features generated from part-of-speech tags, and embedded words. The classifier was trained in part using Sentiment140 data. Docs

Using the sandbox

> I don't want to talk to you no more, you empty headed animal food trough wiper. I fart in your general direction. Your mother was a hamster and your father smelt of elderberries.

Using R

```
library(TextAnalysis)
text<-"I don't want to talk to you no more, you empty headed animal food trough wiper. I fart in your g
someText<-data.frame(id=1, text)
result<-getSentiment(someText)
```

id	text
1	I don't want to talk to you no more, you empty headed animal food trough wiper. I fart in your general direction. You

How'd this do?

Well...

- 0 is highly negative
- 1 is highly positive
- So 0.09 is not very positive
- Do we agree?

Let's try less ambiguous text

```
text=c("I'm filled with happiness and joy",
       "I'm so sad right now and I want to cry")
someMoreText<-data.frame(id=1:2,  text)
result<-getSentiment(someMoreText)
```

id	text	language	score
1	I'm filled with happiness and joy	en	0.8464015
2	I'm so sad right now and I want to cry	en	0.0007791

How'd this do?

Well...

- 0 is highly negative
- 1 is highly positive
- Our happy line (line 1) got 0.85 is not very positive
- Our unhappy line (line 2) got 0 is very negative
- Do we agree?

What might be the problem with our API?

- Text longer than 140 chars
- Idioms
- Humour
- No feedback loop

Exercise

Use the API to get sentiment scores for 5 random lines from the Meaning of Life

- Use the dataset `scriptSpeech`
- Filter to only lines from the Meaning of Life
- Take a sample of 5 of these
- Make the data conform to the spec required by `getSentiment()`
- Get the sentiment scores for each line of speech

Answer

```
scriptSpeech %>%
  filter(Script=="Meaning of Life") %>%
  sample_n(5)  %>%
  select(id=lineid, text=lines) %>%
  getSentiment()
```

id	text	language	score
4087	be quiet! englishmen, you're all so fucking pompous, and none of you have got any balls.	en	0.1321568
2822	yes!	en	0.8335907
5354	hallo. now, don't you worry.	en	0.8141621
5083	oh, we're just fine!	en	0.9740794
3780	excuse me, sir.	en	0.2281185

Sentiment

Sentiment: By the word

- We can analyse sentiment at a variety of levels.
- Individual words is the simplest level.
- Various databases / lexicons available in the tidytext package

Sentiment: Negative or positive

- Bing Liu and co lexicon is commonly used in English

word	sentiment
happy	positive
unhappy	negative

Sentiment: Scale of positivity

- AFINN lexicon is commonly used in English

word	value
happy	3
unhappy	-2

Sentiment: Emotions

- NRC Word-Emotion Association lexicon is commonly used in English

word	sentiment
happy	anticipation
happy	joy
happy	positive
happy	trust
unhappy	anger
unhappy	disgust
unhappy	negative
unhappy	sadness

Lexicon sizes

```
sentiments %>%
  group_by(lexicon) %>%
  summarise(n=n_distinct(word))
```

lexicon	n
afinn	2477
bing	13922
loughran	3917
nrc	6468

How to get sentiment for some words

```
data.frame(word=c("holy","bother", "crucified", "spanking","happy")) %>%
  left_join(get_sentiments("bing")) ->
  bingsentiment
```

word	sentiment
holy	positive
bother	negative
crucified	NA
spanking	NA
happy	positive

Simple overall sentiment

```
bingsentiment %>%
  count(sentiment) %>%
  slice(which.max(n))
```

sentiment	n
positive	2

Exercise

- Use the dataset `scriptWords`
- Filter to only words from the Meaning of Life
- Join the bing sentiment data
- Count how many positive and negative sentiment words there were

Answer

```
scriptWords %>%
  filter(Script=="Meaning of Life") %>%
  left_join(get_sentiments("bing")) %>%
  count(sentiment)
```

```
## Joining, by = "word"
```

sentiment	n
negative	128
positive	124
NA	1760

Recap

Sentiment analysis

- You can use APIs to perform sentiment analysis quickly
  - But it has it's downsides

- You can perform sentiment analysis by scoring words in text using lexicons

dplyr

- General data manipulation package
- Chain commands with `%>%`
- Verbs used so far: `left_join`, `count`, `slice`, `sample_n`, `filter`, `select`

tidytext

- Used for tidy text analysis
- Used so far: `sentiments` table

```
sentiments %>%
  mutate(lexicon="bing") %>%
  bind_rows(
    get_sentiments("afinn") %>%
      mutate(lexicon="afinn") ) %>%
  bind_rows(
    get_sentiments("loughran") %>%
      mutate(lexicon="loughran") ) %>%
  bind_rows(
    get_sentiments("nrc") %>%
      mutate(lexicon="nrc") )->
  sentiments
```

Act 2: Life of Brian

Working with all three lexicons

If you're not sure which sentiment lexicon you want to use, you can use them all and filter later.

```
scriptWords %>%
  inner_join(sentiments) ->
  scoredWords
```

```
## Joining, by = "word"
```

Script	Part	Character	word	sentiment	lexicon	value
Meaning of Life	2	mr. hendy	wonderful	positive	bing	NA
Life of Brian	3	mr. cheeky	ha	NA	bing	2
Hollywood Bowl	2	announcer	accused	fear	nrc	NA
Life of Brian	3	nisus	crucifixion	fear	bing	NA
Holy Grail	3	arthur	risk	anticipation	nrc	NA

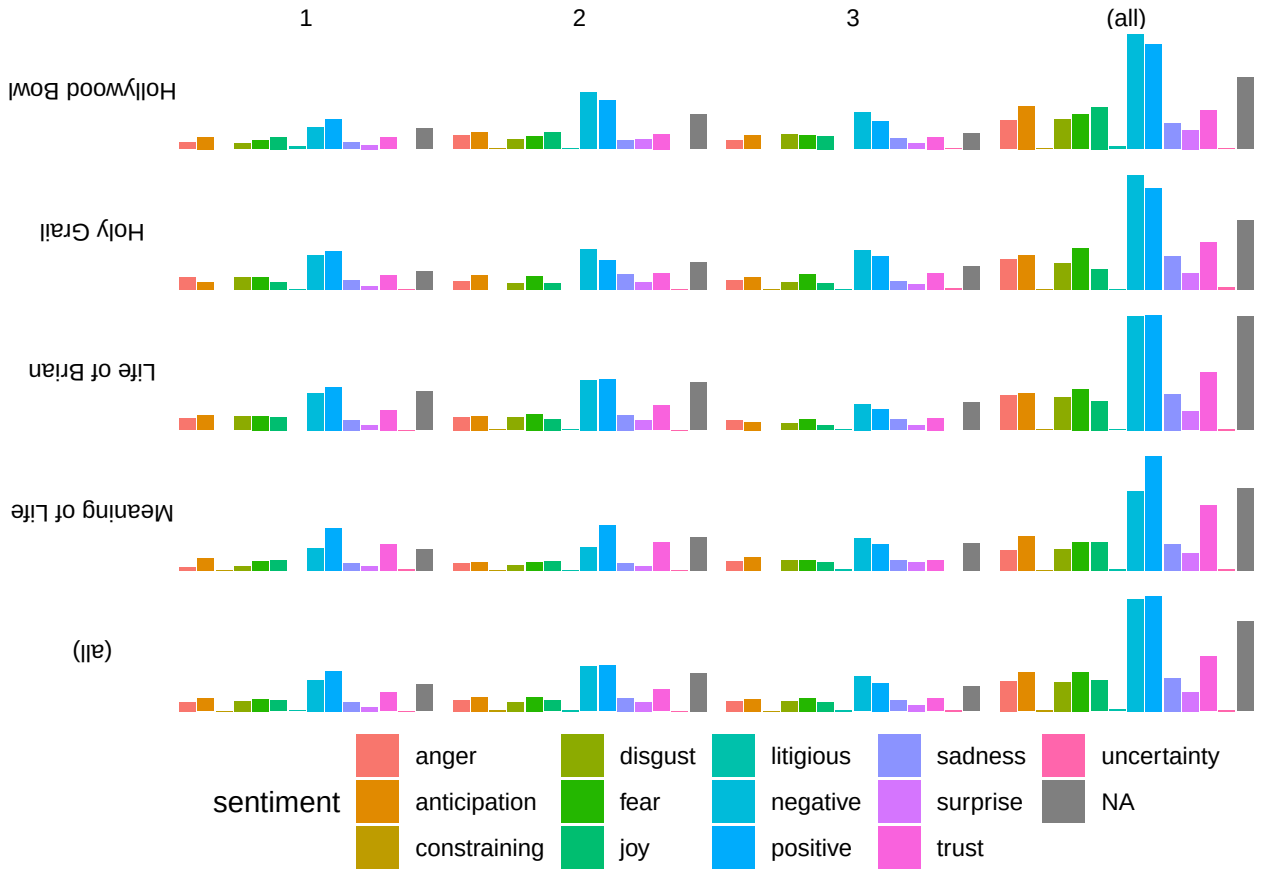
Positive or negative? Data

```
scoredWords %>%
  filter(lexicon=="bing") %>%
  count(Script,Part,sentiment) ->
  bingScoreSummary
```

Script	Part	sentiment	n
Hollywood Bowl	1	anger	44
Hollywood Bowl	1	anticipation	82
Hollywood Bowl	1	disgust	40
Hollywood Bowl	1	fear	60
Hollywood Bowl	1	joy	82
Hollywood Bowl	1	litigious	18

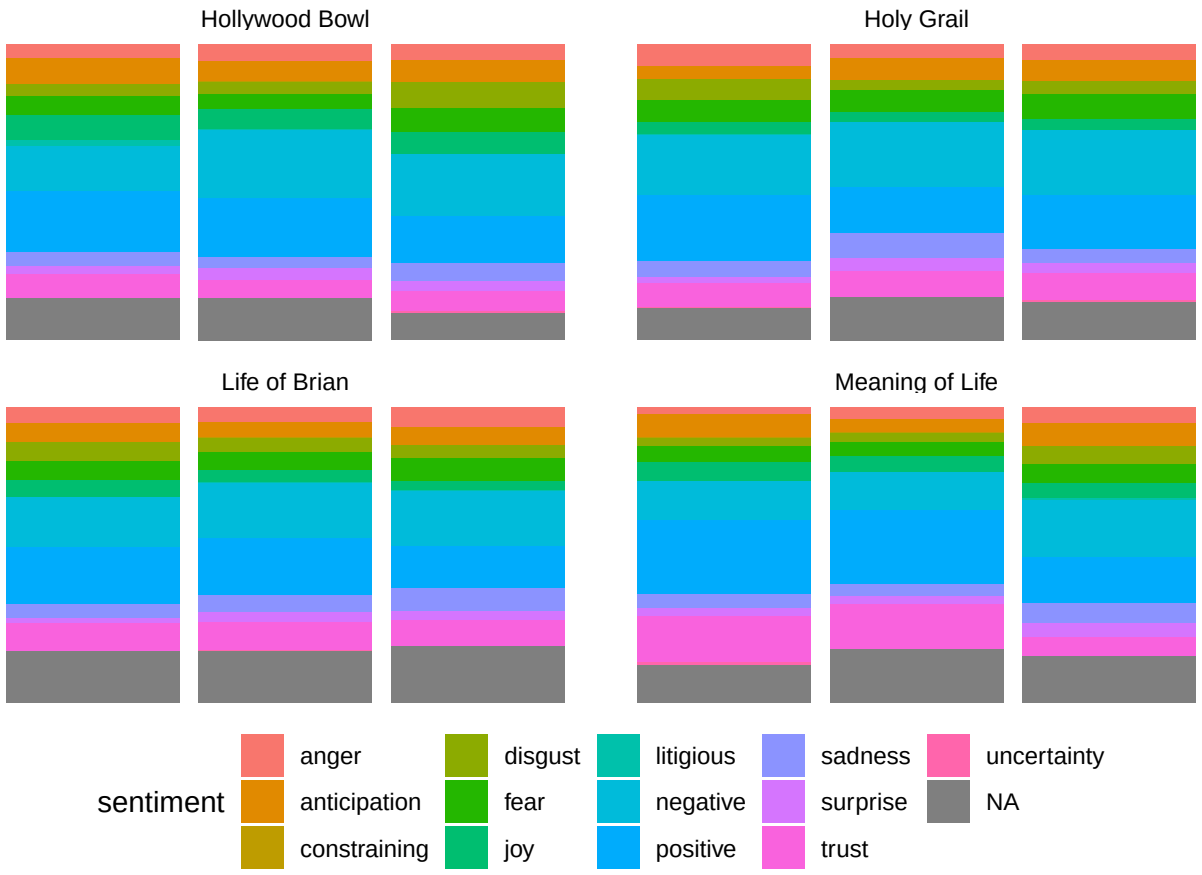
Positive or Negative? Charts

```
ggplot(bingScoreSummary, aes(x=sentiment, y=n, fill=sentiment)) +
  geom_col()+
  facet_grid(Script ~ Part,margins = TRUE, scales = "free",switch = "y")+
  theme_void()+theme(legend.position="bottom")
```



Positive or Negative? Charts

```
ggplot(bingScoreSummary, aes(x=Part, y=n, fill=sentiment)) +
  geom_bar(stat="identity",position = "fill") +
  facet_wrap(~Script)+
  theme_void()+theme(legend.position="bottom")
```



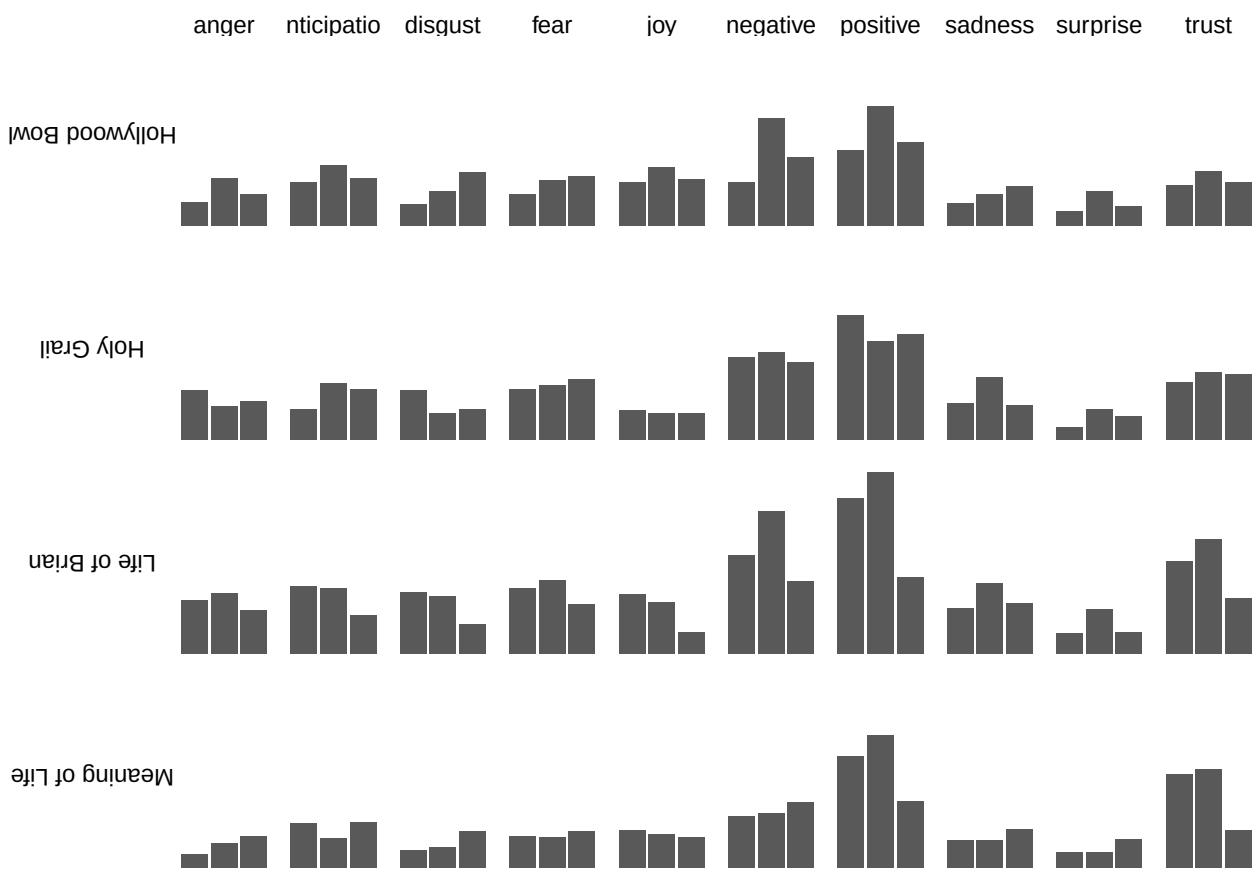
Emotions? Data

```
scoredWords %>%
  filter(lexicon=="nrc") %>%
  count(Script,Part,sentiment) ->
  nrcScoreSummary
```

Script	Part	sentiment	n
Hollywood Bowl	1	anger	22
Hollywood Bowl	1	anticipation	41
Hollywood Bowl	1	disgust	20
Hollywood Bowl	1	fear	30
Hollywood Bowl	1	joy	41
Hollywood Bowl	1	negative	41

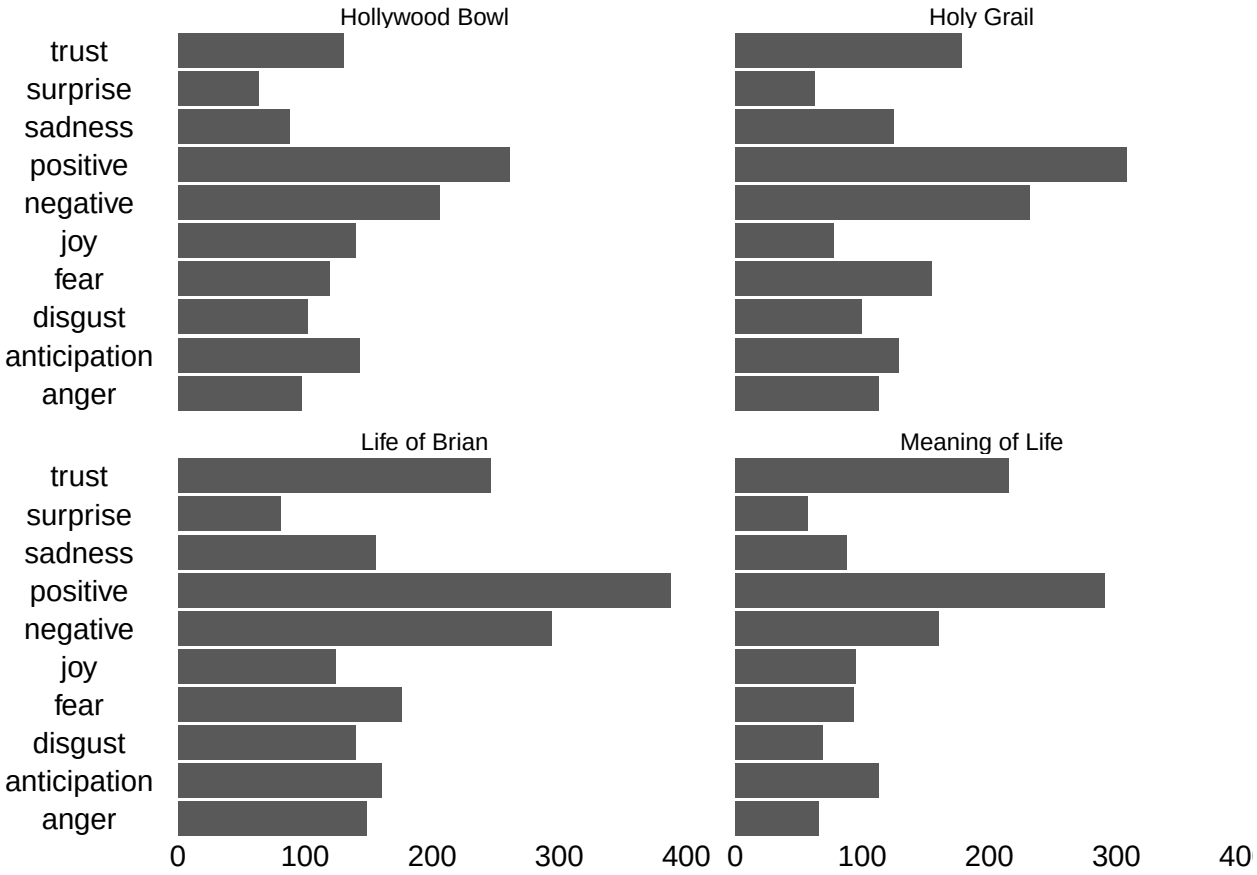
Emotions? Charts

```
ggplot(nrcScoreSummary, aes(x=Part, y=n))+
  geom_col()+
  facet_grid(Script~sentiment,switch = "y")+
  theme_void()
```



Emotions? Charts

```
ggplot(nrcScoreSummary, aes(x=sentiment,y=n))+
  geom_col()+
  coord_flip()+
  facet_wrap(~Script)+
  theme_void()+
  theme(axis.text  = element_text(angle = 0))
```



Exercise

- Get summarised Bing sentiment scores for the top 10 most talkative characters in the Life of Brian
  - Filter `scoredWords` to words with bing sentiment scores
  - Filter to the Life of Brian script
  - Count words by character and sentiment
  - Add an extra column that does the total words by character using `mutate`
  - Ungroup and select the 10 characters with the most words overall using `top_n`
- Use this data to produce a stacked bar chart of sentiment by character
- Use this data to produce a stacked, 100% bar chart of sentiment by character

Answer p1

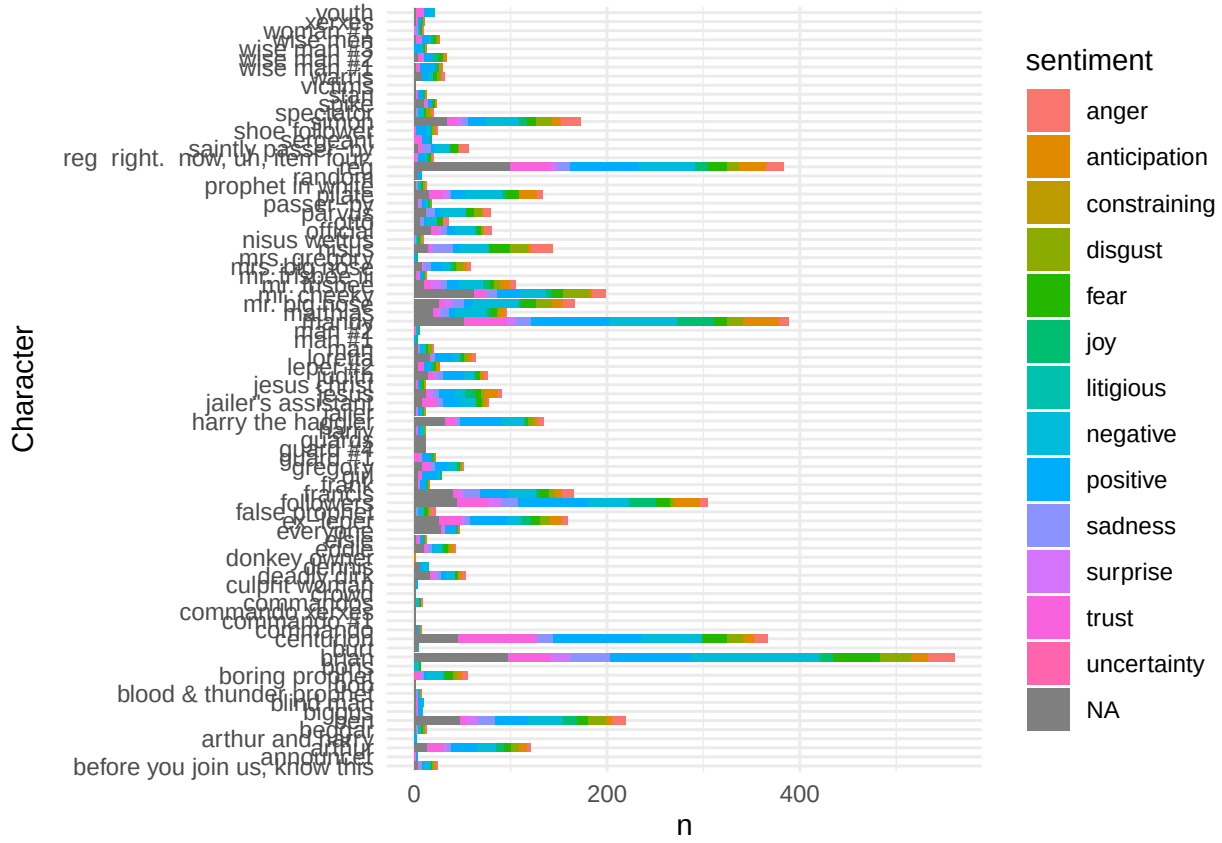
```
scoredWords %>%
  filter(lexicon=="bing") %>%
  filter(Script=="Life of Brian") %>%
  count(Character,sentiment) %>%
  mutate(TotalWords=sum(n)) %>%
  ungroup() %>%
  top_n(20,TotalWords) ->
  lifeofbriansentiment
```

Character	sentiment	n	TotalWords
before you join us, know this	anger	2	5412
before you join us, know this	anticipation	2	5412

Character	sentiment	n	TotalWords
before you join us, know this	disgust	2	5412
before you join us, know this	fear	2	5412
before you join us, know this	negative	8	5412
before you join us, know this	positive	1	5412

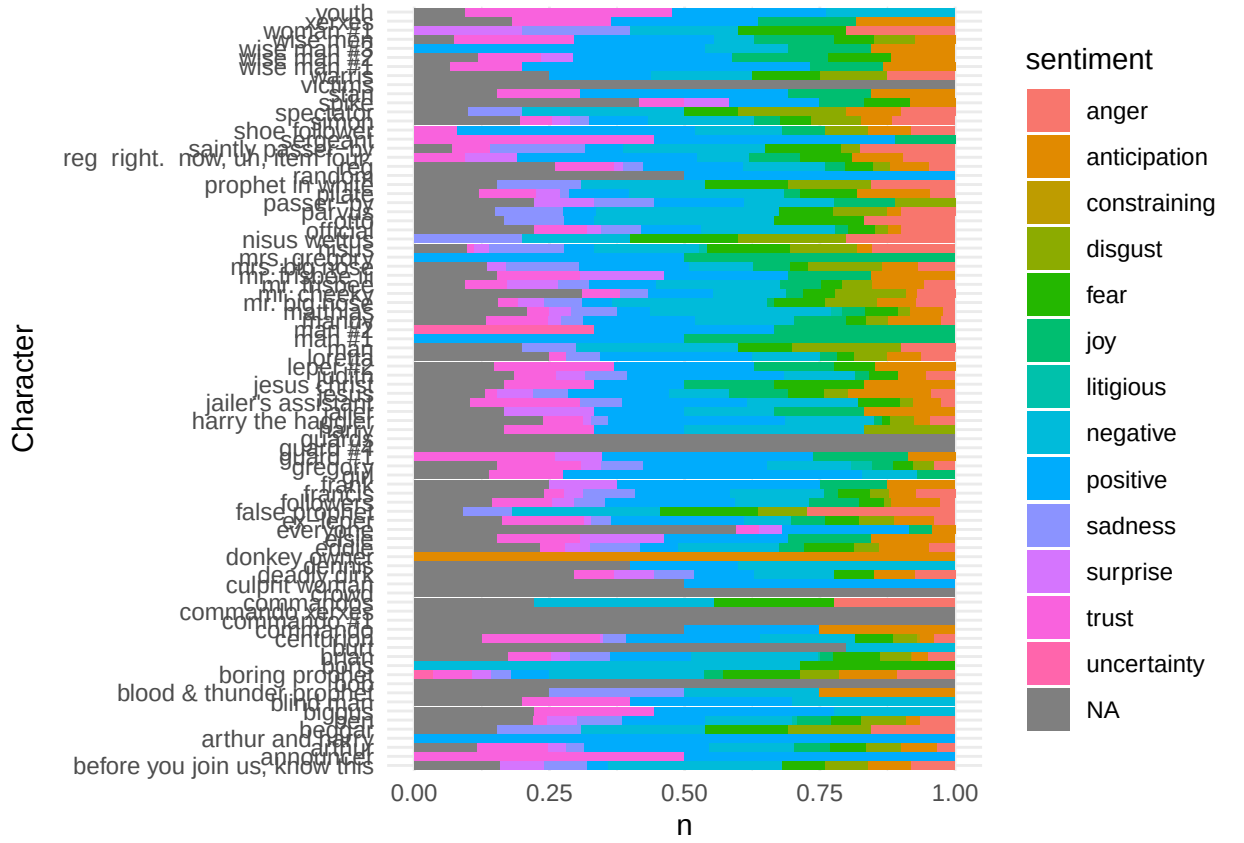
Answer p2

```
ggplot(lifeofbriansentiment, aes(x=Character,y=n,fill=sentiment))+
  geom_col() +
  theme_minimal()+
  coord_flip()
```



Answer p3

```
ggplot(lifeofbriansentiment, aes(x=Character,y=n,fill=sentiment))+
  geom_bar(stat="identity",position = "fill") +
  theme_minimal()+
  coord_flip()
```



And now for something completely different...

The tidy text concept

We've been working with tabular data but until pretty darn recently text processing was pretty messy. Enter Julia Silge with the package `tidytext` and the fab book Tidy Text Mining.

Let's look at how we've been applying some of the tidy text principles.

Need scripts in a tidy format

Get each script / document as a row in a data.frame.

- We'll look at webscraping later but for now, just inspect `getScriptData`.
- Key part is fetching data
- Add IDs with `row_number()`

```
getScriptURLs() %>%
  purrr::by_row(getScript) %>%
  dplyr::bind_rows()
```

```
## # A tibble: 1 x 6
##   showid scriptid Script   Part  URL             ScriptText
##    <int>    <int> <chr>    <chr> <chr>           <chr>
## 1      2        6 Holy Gr~ 3     http://www.mon~ "\r\n\t\r\n\t\r\n\r\n\r\n\r\n~
```

Housekeeping

- Inevitably, text isn't very clean, especially after web scraping! You need to perform cleanup.
- Make sure to add IDs to entity observations as you go along
  - IDs for scripts
  - IDs for lines
  - etc
  - With scripts, we need to remove extra content like actions, cast list, and preamble
  - Inspect `getScriptData`, `getScriptLines`, `getScriptSpeech`

Decomposing to tokens

We need to transform overall text to smaller tokens, whether paragraphs, lines, n-grams, or words.

- Use `unnest_tokens()` to perform the decomposition
- Inspect `getScriptLines` to see decomposition to lines

```
getScriptData() %>%
  unnest_tokens(lines, ScriptText, token = "lines", collapse = TRUE)
```

showid	scriptid	Script	Part	URL	lines
1	1	Hollywood Bowl	1	http://www.montypython.net/../hbowlmm1.php	
1	1	Hollywood Bowl	1	http://www.montypython.net/../hbowlmm1.php	
1	1	Hollywood Bowl	1	http://www.montypython.net/../hbowlmm1.php	monty python live at the h
1	1	Hollywood Bowl	1	http://www.montypython.net/../hbowlmm1.php	sit on my face
1	1	Hollywood Bowl	1	http://www.montypython.net/../hbowlmm1.php	self-wrestling
1	1	Hollywood Bowl	1	http://www.montypython.net/../hbowlmm1.php	never be rude to an arab p

Stop words

There are many words that are very common connectors in text. As they're so common, they add little to an analysis so these are often removed.

word	lexicon
i	snowball
given	SMART
themselves	SMART
perhaps	onix
very	onix

Removing stopwords

```
getScriptSpeech() %>%
  unnest_tokens(word, lines) %>%
  anti_join(stop_words)
```

Script	Part	URL	Character	word
Hollywood Bowl	3	http://www.montypython.net/../hbowlmm3.php	wife	landing
Hollywood Bowl	2	http://www.montypython.net/../hbowlmm2.php	eric idle	conditioned
Life of Brian	2	http://www.montypython.net/../brianmm2.php	matthias	legs

Script	Part	URL	Character	word
Life of Brian	1	http://www.montypython.net/../brianmm1.php	brian	struggling
Meaning of Life	3	http://www.montypython.net/../meaningmm3.php	man	uh

Exercise

What were the most prevalent stop words by script?

- Start with the `scriptLines` dataset
- Filter it to where a line is Speech and not an Action
- Use `unnest_tokens` to decompose lines to words
- Use `semi_join` with `stop_words` to get the overlap
- Count by script and word
- Return top 5 by script

Answer

```
scriptLines %>%
  filter(Speech, !Action) %>%
  unnest_tokens(word, lines) %>%
  semi_join(stop_words) %>%
  count(Script, word) %>%
  group_by(Script) %>%
  top_n(5) ->
  topStopWords
```

```
## Joining, by = "word"
```

```
## Selecting by n
```

Script	word	n
Hollywood Bowl	a	184
Hollywood Bowl	and	161
Hollywood Bowl	i	156
Hollywood Bowl	the	308
Hollywood Bowl	you	188
Holy Grail	a	163

Recap

Sentiment analysis

- You can use APIs to perform sentiment analysis quickly
  - But it has it's downsides
- You can perform sentiment analysis by scoring words in text using lexicons
- You can use the way a text is broken down to see sentiment change over the course of the text
- You need to decompose bigger documents into smaller components
- You should remove stop words

dplyr

- General data manipulation package

- Chain commands with `%>%`
- Verbs used so far: `left_join`, `count`, `slice`, `sample_n`, `filter`, `select`, `top_n`, `semi_join`, `anti_join`, `mutate`

tidytext

- Used for tidy text analysis
- Used so far: `sentiments` table, `get_sentiments`, `unnest_tokens`

ggplot2

- Used for charting
- Made basic and facetted plots with different types of series

Act 3: The Holy Grail

Beyond simple sentiment

Sentiment analysis is hip and everybody's doing it on Twitter but there's a lot more fish in the sea!

tf-idf

tf-idf is how important a word is to a given text, relative to it's presence in other documents

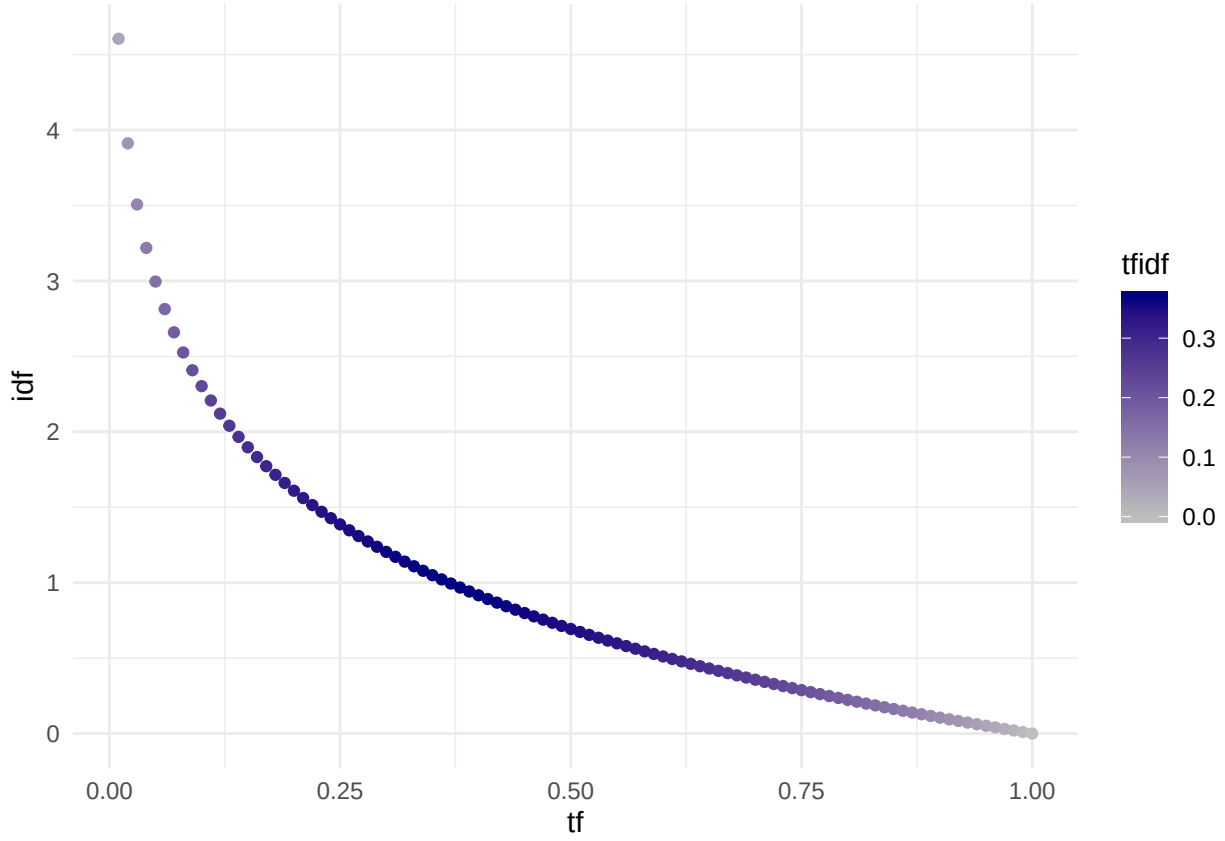
- Term frequency (tf) tells you the proportion of words in a document that are a given word
- Inverse document frequency (idf) is the log of the count of all documents vs. the count of docs with the term in

```
wordfreqindoc     <- 10
wordsindoc        <- 100
documentswithword <- 1
totaldocuments    <- 2

tf <- wordfreqindoc/wordsindoc
idf <- log(totaldocuments/documentswithword)
`tf-idf` <- tf*idf
`tf-idf`
```

```
## [1] 0.06931472
```

tf-idf



Using tf-idf

We can calculate tf_idf using tidytext for our scripts.

```
scriptAllWords %>%
  count(Script, word) ->
  wordfreqs

wordfreqs %>%
  bind_tf_idf(word, Script, n) ->
  wordtfidf
```

Script	word	n	tf	idf	tf_idf
Holy Grail	times	2	0.0003113	0.2876821	0.0000896
Holy Grail	seem	1	0.0001556	0.6931472	0.0001079
Holy Grail	bother	2	0.0003113	1.3862944	0.0004315
Holy Grail	three	17	0.0026459	0.0000000	0.0000000
Holy Grail	aramathea	2	0.0003113	1.3862944	0.0004315

Using tf-idf

This can tell us about important people or topics in each script.

```
wordtfidf %>%
  group_by(Script) %>%
  top_n(5, tf_idf)
```

Script	word	n	tf	idf	tf_idf
Holy Grail	grail	32	0.0049805	1.386294	0.0069045
Holy Grail	ni	32	0.0049805	1.386294	0.0069045
Holy Grail	launcelot	22	0.0034241	1.386294	0.0047468
Holy Grail	witch	20	0.0031128	1.386294	0.0043153
Holy Grail	castle	19	0.0029572	1.386294	0.0040995

Exercise

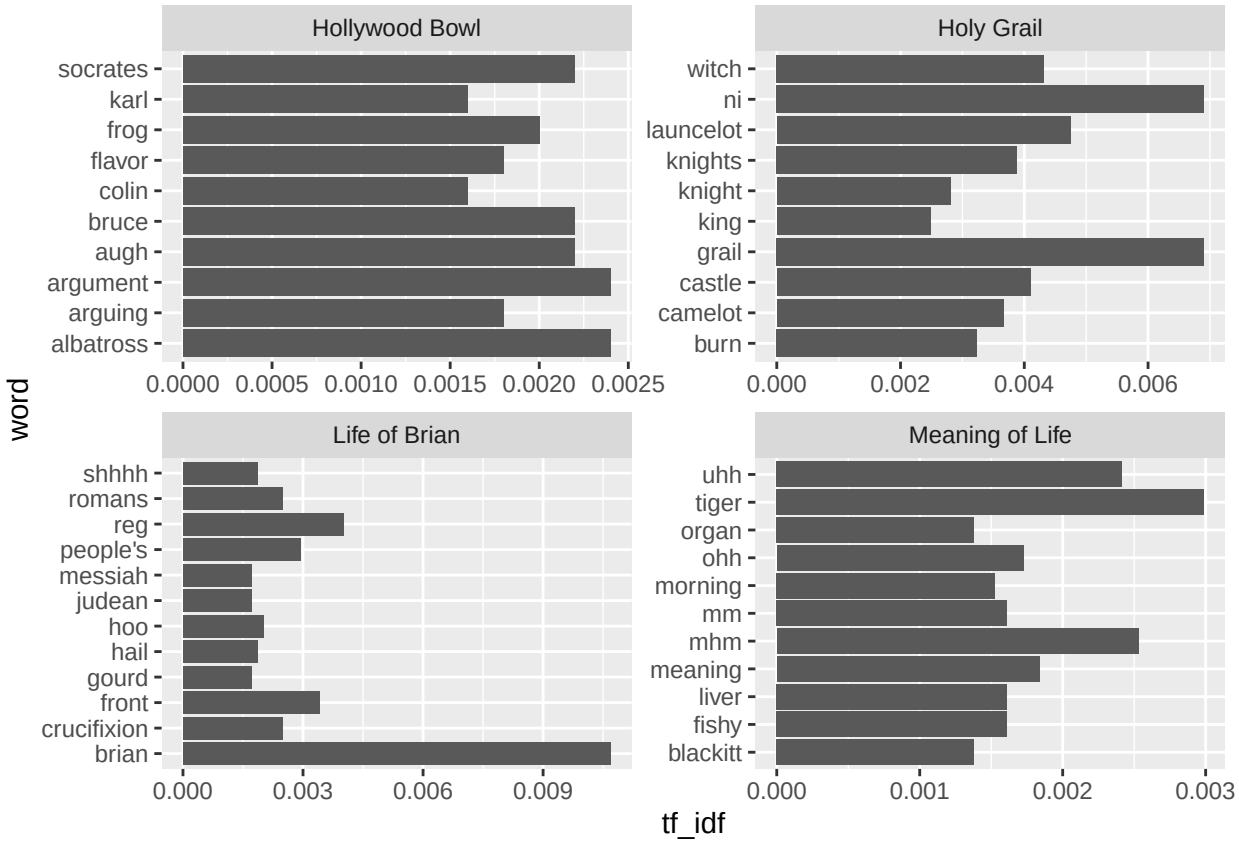
Plot the 10 most important words per script

- Calculate term frequency by script
- Use `bind_tf_idf` to calculate tf-idf for each term in each script
- Take the top 10 terms by tf-idf for each script
- Use this dataset to construct a plot for each script

Answer

```
scriptAllWords %>%
  count(Script, word) %>%
  bind_tf_idf(word, Script, n) %>%
  group_by(Script) %>%
  top_n(10, tf_idf) ->
  top10words

ggplot(top10words, aes(x=word, y=tf_idf)) +
  geom_col() +
  coord_flip()+
  facet_wrap(~Script, scales="free")
```



Beyond single-word sentiment

We mentioned in the Meaning of Life about n-grams! Let's take a look at what we can do with n-grams.

Making n-grams

```
scriptSpeech %>%
  unnest_tokens(bigram, lines, token = "ngrams", n=2) ->
  bigrams
```

Script	Part	URL	Character	bigram
Holy Grail	1	http://www.montypython.net/../grailmm1.php	bedevere	and that
Holy Grail	1	http://www.montypython.net/../grailmm1.php	bedevere	that my
Holy Grail	1	http://www.montypython.net/../grailmm1.php	bedevere	my liege
Holy Grail	1	http://www.montypython.net/../grailmm1.php	bedevere	liege is
Holy Grail	1	http://www.montypython.net/../grailmm1.php	bedevere	is how
Holy Grail	1	http://www.montypython.net/../grailmm1.php	bedevere	how we

Removing n-grams containing stop words

```
bigrams %>%
  tidyr::separate(bigram, c("word1", "word2"), sep = " ") %>%
  anti_join(stop_words, c("word1"="word")) %>%
  anti_join(stop_words, c("word2"="word")) %>%
```

```
  tidyr::unite(bigram, word1, word2, sep = " ") ->
  meaningfulBigrams
```

Script	Part	URL	Character	bigram
Holy Grail	1	http://www.montypython.net/../grailmm1.php	all	NA NA
Holy Grail	1	http://www.montypython.net/../grailmm1.php	arthur	bloody peasant
Holy Grail	1	http://www.montypython.net/../grailmm1.php	arthur	NA NA
Holy Grail	1	http://www.montypython.net/../grailmm1.php	arthur	sacred quest
Holy Grail	1	http://www.montypython.net/../grailmm1.php	arthur	holy grail
Holy Grail	1	http://www.montypython.net/../grailmm1.php	arthur	sir knight

Most common n-grams

```
meaningfulBigrams %>%
  count(Script, bigram) %>%
  group_by(Script) %>%
  top_n(5, n) ->
  topBigrams
```

Script	bigram	n
Holy Grail	haw haw	9
Holy Grail	holy grail	13
Holy Grail	NA NA	107
Holy Grail	ni ni	8
Holy Grail	sir launcelot	12

Working with negations

One of the biggest problems with single word sentiment analysis is a lack of negation.

Including these in n-grams allows us to look at where misclassifications happen.

Negations are stop words

Negations are stop words so we need to remove negations from our list of stop words.

```
stop_words %>%
  anti_join(negations) ->
  new_stop_words
```

```
## Joining, by = "word"
```

Remove stop words

```
bigrams %>%
  tidyr::separate(bigram, c("word1", "word2"), sep = " ") %>%
  anti_join(new_stop_words, c("word1"="word")) %>%
  anti_join(stop_words, c("word2"="word")) ->
  incNegationsBigrams
```

Negated words

```
incNegationsBigrams %>%
  filter(word1 %in% negations$word) ->
  negatedWords
```

Script	Part	URL	Character	word1	word2
Holy Grail	2	http://www.montypython.net/../grailmm2.php	father	doesn't	leave
Holy Grail	1	http://www.montypython.net/../grailmm1.php	guard #1	not	carry
Holy Grail	1	http://www.montypython.net/../grailmm1.php	god	don't	grovel
Holy Grail	3	http://www.montypython.net/../grailmm3.php	galahad	no	yel
Holy Grail	2	http://www.montypython.net/../grailmm2.php	head knight	never	pass
## Negate sen	timent				

```
negatedWords %>%
  inner_join(get_sentiments("bing"),c("word2"="word")) %>%
  mutate(sentiment=ifelse(sentiment=="positive","negative","positive")) %>%
  select( everything(),word=word2,-word1) ->
  negatedSentiment
```

showid	scriptid	lineid	characterid	Script	Part	URL	Spee
1	1	5581	1	Hollywood Bowl	1	http://www.montypython.net/../hbowlmm1.php	TRU
1	1	5594	1	Hollywood Bowl	1	http://www.montypython.net/../hbowlmm1.php	TRU
1	2	2574	1	Hollywood Bowl	2	http://www.montypython.net/../hbowlmm2.php	TRU
1	2	2574	1	Hollywood Bowl	2	http://www.montypython.net/../hbowlmm2.php	TRU
1	2	3400	1	Hollywood Bowl	2	http://www.montypython.net/../hbowlmm2.php	TRU
1	2	3700	1	Hollywood Bowl	2	http://www.montypython.net/../hbowlmm2.php	TRU

Using tf-idf on n-grams

We can calculate tf_idf on n-grams too

```
meaningfulBigrams %>%
  count(Script, bigram) ->
  bigramfreqs

bigramfreqs %>%
  bind_tf_idf(bigram, Script, n) ->
  bigramtfidf
```

Script	bigram	n	tf	idf	tf_idf
Holy Grail	stay ere	1	0.001692	1.386294	0.0023457
Holy Grail	thirds majority	1	0.001692	1.386294	0.0023457
Holy Grail	idiom sir	1	0.001692	1.386294	0.0023457
Holy Grail	outrageous accent	1	0.001692	1.386294	0.0023457
Holy Grail	witch witch	1	0.001692	1.386294	0.0023457

Using tf-idf on n-grams

Are the most important terms the same as the most common ones?

```
bigramtfidf %>%
  group_by(Script) %>%
  top_n(5, tf_idf)
```

Script	bigram	n	tf	idf	tf_idf
Holy Grail	holy grail	13	0.0219966	1.386294	0.0304938
Holy Grail	sir launcelot	12	0.0203046	1.386294	0.0281481
Holy Grail	haw haw	9	0.0152284	1.386294	0.0211111
Holy Grail	ni ni	8	0.0135364	1.386294	0.0187654
Holy Grail	arthur king	7	0.0118443	1.386294	0.0164197
Holy Grail	sir galahad	7	0.0118443	1.386294	0.0164197
Holy Grail	sir robin	7	0.0118443	1.386294	0.0164197

Why bother?

How many of us search for just one word topics? We can use tf-idf to narrow down documents that are likely matches to our search term but we need to check tf-idf for our multi-word term.