TED Talks Popularity Predictor

HardvardX Professional Certificate in Data Science - Capstone Project 2 - CYO

Joao Rodrigues

17 June 2019

1. Executive summary

This report documents the analytical approach, modelling results and proposed next steps for the HarvardX Data Science Capstone CYO Project. This project analyses the TED Talks dataset and identifies predictors of the popularity of talks. This has commercial application in media and content programming, e.g. predicting the popularity of speakers and their topics on broadcast shows of various formats (television, podcast, radio). Some of the dimensions modelled to predict popularity include: the topic of the talks (tags), ratings, sentiment analysis on the transcripts, talk duration and speed of delivery (words/minute), and number of comments. Predictive models were built using knn and random forest methodologies. The modelling approach predicts the quartile of views that a given TED Talk topic is expected to fall into. Results are segmented into 2 training datasets - one set that includes parameters only available after the talk has screened (e.g. comments, ratings) and a second set that only models for parameters known at date of publication (e.g. topic of talks, sentiment analysis of the transcript, etc.)

2. Objectives of this analysis

This project sets out to demonstrate an appropriate approach to data exploration, and application of select machine learning algorithms to predict popularity of talks from the TED Talks. Specifically, this project demonstrates:

- Data pre-processing (especially text wrangling to extract subject topics for talks, ratings data, and sentiment analysis on the transcripts)
- Exploratory data analysis to dimension the dataset and test for various correlations to the number of views that a talk attracts
- The application of machine learning algorithms to predict, based on relevant factors, the popularity of talks (as measured by the quartile of views that a talk falls into)

3. Methodology

3.1. Preprocessing

3.1.1. Introduction to the TED Talks datasets - main data set and transcripts

This project explores 2 primary datasets, `ted_main_data` and `ted_transcript_data`. As per the below code, the ted_main_data dataset is a matrix with 2550 observations and 17 variables.

```
#display ted_main_data and dimensions
glimpse(ted_main_data)
```

```
## Observations: 2,550
## Variables: 17
## $ comments           <int> 4553, 265, 124, 200, 593, 672, 919, 46, 852...
## $ description        <fct> "Sir Ken Robinson makes an entertaining and...
## $ duration           <int> 1164, 977, 1286, 1116, 1190, 1305, 992, 119...
## $ event              <fct> TED2006, TED2006, TED2006, TED2006, TED2006...
## $ film_date          <int> 1140825600, 1140825600, 1140739200, 1140912...
## $ languages          <int> 60, 43, 26, 35, 48, 36, 31, 19, 32, 31, 27,...
## $ main_speaker       <fct> Ken Robinson, Al Gore, David Pogue, Majora ...
## $ name               <fct> "Ken Robinson: Do schools kill creativity?"...
## $ num_speaker        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ published_date     <int> 1151367060, 1151367060, 1151367060, 1151367...
## $ ratings            <fct> "[{'id': 7, 'name': 'Funny', 'count': 19645...
## $ related_talks      <fct> "[{'id': 865, 'hero': 'https://pe.tedcdn.co...
## $ speaker_occupation <fct> "Author/educator", "Climate advocate", "Tec...
## $ tags               <fct> "['children', 'creativity', 'culture', 'dan...
## $ title              <fct> "Do schools kill creativity?", "Averting th...
## $ url                <fct> https://www.ted.com/talks/ken_robinson_says...
## $ views              <int> 47227110, 3200520, 1636292, 1697550, 120058...
```

The `ted_transcript_data` dataset is a matrix with 2467 observations and 2 variables

```
#display ted_transcripts_data and dimensions
glimpse(ted_transcripts_data)
```

```
## Observations: 2,467
## Variables: 2
## $ transcript <chr> "Good morning. How are you?(Laughter)It's been grea...
## $ url        <chr> "https://www.ted.com/talks/ken_robinson_says_school...
```

Given the intent to use transcripts analysis as a core part of the prediction algorithm later, we slightly reduce the `ted_main_data` observations to ensure we have transcripts for all observations, using the `semi_join()` coding below . . .

```
# ensure we have transcripts for all
ted_main_data <- semi_join(ted_main_data, ted_transcripts_data, by = "url")
```

. . . and confirm that all these remaining observations (2464) are complete cases.

```
# remove duplicate entries in transcripts
ted_transcripts_data <- distinct(ted_transcripts_data)

#sort both by url
ted_transcripts_data <- ted_transcripts_data[order(match(ted_transcripts_data$url, ted_main_data$url)), ]

# check for complete cases on all data
sum(complete.cases(ted_main_data))-nrow(ted_main_data)
```

```
## [1] 0
```

3.1.2. Pre-processing to separate tags by TED Talk into a matrix

The `tags` field in the main dataset records the multiple topics that a specific talk relates to. Text wrangling per the below code separates the unique tag words into a vector and then collapses to the unique tag names with the `ndistinct()` dplyr function per below:

```
# calc max_no_of_tags
tags_tidy <- ted_main_data$tags %>%
  as.character() %>%
  strsplit(., ", ") %>%
  unlist() %>%
  gsub("[[:punct:]]", "", .) %>%
  enframe()

#n_distinct(tags_tidy)
tags_distinct <- tags_tidy %>% group_by(value) %>% summarize(n = n()) %>% arrange(desc(n))
#n_distinct(tags_distinct)
tags_matrix <- matrix(nrow = nrow(ted_main_data), ncol = nrow(tags_distinct))
```

A `tags_matrix` is then created and each TED Talk observation has a boolean flag indicating whether the specific talk covers that tagged topic or not.

```
for (i in 1:nrow(ted_main_data)) {
  for (j in 1:nrow(tags_distinct)){
        tags_matrix[i, j] <- str_detect(as.character(ted_main_data$tags[i]), paste("\'", as.character(tags_distinct[j, 1]),
"\'", sep = ""))
  }
}
colnames(tags_matrix) <- unlist(tags_distinct[1:nrow(tags_distinct), "value"])
```

This matrix of tags, of dimensions 2464 rows and 416 distinct tags, will be column bound to the `ted_main_data dataset` later for further predictive analysis.

```
glimpse(tags_matrix)
```

```
##  logi [1:2464, 1:416] FALSE TRUE TRUE FALSE FALSE FALSE ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : Named chr [1:416] "technology" "science" "global issues" "culture" ...
##   .. ..- attr(*, "names")= chr [1:416] "value1" "value2" "value3" "value4" ...
```

3.1.3. Separate ratings into unique rating names and counts

The ratings data in `ted_main_data` is factorized vector of concatenated strings that contain multiple ratings categories and counts of ratings per category. The text wrangling strips out header information and punctuation, and transforms this dataset into a matrix comprised of a unique id for the rating type, the name of the rating type, and the count of that specific ratings. This involves a few processing steps to:

- extract a 2464 x 14 column list of the count of ratings scores for each TED Talk, in a string of format “id number, description, number of ratings”
- a function is defined to transform the string of ratings scores into a dataframe with `id`, `desc`, `no_of_ratings`
- the function is applied to the long ratings listing (in string format) and converts to a tidy dataframe
- finally we define a matrix of 2464 rows x 14 distinct ratings types in columns and populate with the ratings data.

This resultant matrix has the following structure:

```
glimpse(ted_main_data_ratings_matrix)
```

```
##  num [1:2464, 1:14] 19645 544 964 59 1390 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:14] "Funny" "Beautiful" "Ingenious" "Courageous" ...
```

. . . and will be used to test for predictive power in the exploratory data analysis and model training

This pre-processing of the ratings data in `ted_main_data` is achieved with the following code chunk:

```
#extract a 14 row x nrow(ted_main_data) col list of the the count of ratings scores for each ted talk, in a string of forma
t "id number, description, number of ratings"

list_summary_ratings <- sapply(ted_main_data$ratings[1:nrow(ted_main_data)], function(x) {
    x <- x %>%
    str_replace(., "^\\[", "") %>%
    str_replace(., "\\]$", "") %>%
    str_replace_all(., "\\'id\': |\'name\': |\'count\': ", "") %>%
    str_replace_all(., "[[:blank:]]", "") %>%
    str_split(., "\\}") %>%
    unlist() %>%
    str_replace(., ",\\{|\\{", "")
})
list_summary_ratings <- list_summary_ratings %>% t()
rownames(list_summary_ratings) <- ted_main_data$url[1:nrow(ted_main_data)]

#create a function to transform the string of ratings scores into a dataframe with id, desc, no of ratings
rating_df <- function(x) {
      temp_str_vector <- unlist(str_split(x, ","))
      data.frame (id = as.integer(temp_str_vector[1]),
      rating_type = str_replace_all(temp_str_vector[2], "[[:punct:]]", ""),
      no_of_ratings = as.numeric(temp_str_vector[3]))
    }

#initialise a blank dataframe
ted_talks_ratings_all <- data.frame(id = integer(),
                                    rating_type = character(),
                                    no_of_ratings = numeric())

#create a loop that applies the function to the ratings listing (string format) and converts to a long dataframe with all t
he data
for (i in 1:nrow(ted_main_data)) { #Loop for each ted talk
 for (j in 1:length(list_summary_ratings[1, ])-1) #Loop for each rating entry per ted talk
  {
  ted_talks_ratings_all <- rbind(ted_talks_ratings_all, rating_df(list_summary_ratings[i, j]))
  }
}

rating_types <- levels(ted_talks_ratings_all$rating_type)
# after interim processing, restore punctuation to one of the key ratings categories
rating_types[11] <- "Jaw-dropping"

#create a matrix of nrow(ted_main_data) rows x n_distinct_rating types columns and populate with the views data
ted_main_data_ratings_matrix <- matrix(nrow = nrow(ted_main_data), ncol = n_distinct(ted_talks_ratings_all$rating_type))
colnames(ted_main_data_ratings_matrix) <- rating_types

for (i in 1:nrow(ted_main_data)) {
  for (j in 1:n_distinct(ted_talks_ratings_all$rating_type))
       {
         #add code here to populate matrix of rating number by rating type from list_summary_ratings
         ted_main_data_ratings_matrix[i, j] <- ifelse(str_detect(ted_main_data$ratings[i], rating_types[j]),
                                                      as.numeric(str_extract(str_extract(ted_main_data$ratings[i], paste(
"\'name\': \'", rating_types[j], "\', \'count\': \\d+\\}", sep="")), "\\d+")),
                                                      0)}}
```

3.1.4. Convert time stamps to UTC format

The date fields (`film_date` and `published_date`) are converted from a UNIX timestamp into UTC format using the `lubridate` package as follows:

```
# convert time stamps into UTC format
library(lubridate)
ted_main_data$film_date <- as.Date(structure(ted_main_data$film_date, class = c("POSIXct", "POSIXt")))
ted_main_data$published_date <- as.Date(structure(ted_main_data$published_date, class = c("POSIXct", "POSIXt")))
```

3.1.5. Sentiment analysis on transcripts

Finally, in terms of pre-processing, sentiment analysis is performed on transcripts using 2 lexicons. This is achieved with the following steps:

- Unnesting individual words in the transcript, removing commonly used stop words that won't influence sentiment. Before subsetting further for sentiment words, the highest frequency words (after removing stop words) are as follows:

```
#display top word counts
tidy_transcript %>%
  group_by(word) %>%
  tally %>%
  arrange(desc(n)) %>%
  head(10)
```

```
## # A tibble: 10 x 2
##    word         n
##    <chr>    <int>
##  1 people   18969
##  2 laughter 10312
##  3 time     10225
##  4 world     9401
##  5 life      5859
##  6 applause  5525
##  7 lot       5232
##  8 day       4381
##  9 called    3996
## 10 percent   3833
```

- Applying the `nrc` lexicon first, we inner join the `nrc` lexicon with that of the transcript data to yield a matrix with the TED Talk URL as the unique identifier, and the `nrc` sentiment count. `NRC` classifies certain words into 10 emotional groupings, e.g. anger, anticipation, disgust. The results of this sentiment analysis are shown below. This will furthermore allow testing of talk popularity based on relative mix of sentiments in the talk and the absolute count of emotive language.

```
# nrc lexicon
# inner join the sentiment associated with the matched words from the nrc lexicon
tidy_transcript_nrc_words <- tidy_transcript %>%
  inner_join(get_sentiments("nrc"), by = "word")

# summarise the sentiment data by ted talk, with URL as the unique identifier
ted_transcript_sentiment_nrc <- tidy_transcript_nrc_words %>%
  group_by(url, sentiment) %>%
  summarize(n = n()) %>%
  spread(key = sentiment, value = n)

glimpse(ted_transcript_sentiment_nrc)
```

```
## Observations: 2,464
## Variables: 11
## Groups: url [2,464]
## $ url          <chr> "https://www.ted.com/talks/9_11_healing_the_mothe...
## $ anger        <int> 15, 24, 2, 1, 4, NA, 15, 36, 15, 5, 11, 14, 18, 3...
## $ anticipation <int> 13, 30, 6, 3, 10, NA, 19, 22, 42, 11, 20, 14, 10,...
## $ disgust      <int> 5, 16, 2, NA, 2, NA, 7, 18, 7, 2, 8, 8, 14, 2, 15...
## $ fear         <int> 20, 29, 4, 2, 4, NA, 21, 46, 21, 3, 12, 16, 30, 5...
## $ joy          <int> 14, 32, 6, 4, 11, 2, 21, 19, 30, 7, 13, 17, 11, 1...
## $ negative     <int> 28, 42, 7, 3, 5, NA, 41, 61, 33, 8, 19, 36, 34, 9...
## $ positive     <int> 24, 71, 22, 7, 24, 2, 59, 52, 71, 35, 42, 58, 52,...
## $ sadness      <int> 15, 16, 4, 1, 4, 1, 25, 37, 18, 3, 10, 14, 20, 6,...
## $ surprise     <int> 7, 16, 3, 3, 4, 1, 8, 10, 15, 4, 10, 11, 9, 6, 18...
## $ trust        <int> 16, 36, 9, 1, 12, 1, 37, 43, 41, 15, 14, 28, 22, ...
```

- Word cloud analysis of the `tidy_transcript_nrc_words` visualises the highest frequency words driving the sentiment analysis.

```
#plot a wordcloud of the nrc sentiment words
#install.packages(c("tm", "SnowballC", "wordcloud", "RColorBrewer", "RCurl", "XML"))
library(tm)
library(SnowballC)
library(wordcloud)
library(RColorBrewer)
library(RCurl)

wordcloud(tidy_transcript_nrc_words$word, min.freq = 1, max.words = 50, random.order = FALSE, random.color = FALSE, colors
 = "blue4")
```

- Applying the `bing` lexicon next, this categorises a set of words into positive or negative sentiment, so we can also calculate the net positive or negative sentiment score as well as the sum of positive and negative sentiment words (again, a measure of emotive language in a given talk).

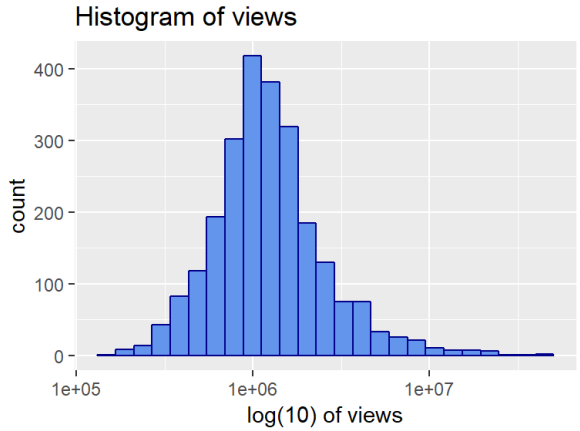
```
## Observations: 2,456
## Variables: 3
## Groups: url [2,456]
## $ url      <chr> "https://www.ted.com/talks/9_11_healing_the_mothers_w...
## $ negative <int> 20, 46, 5, 2, 5, 43, 46, 21, 14, 18, 29, 37, 7, 42, 1...
## $ positive <int> 10, 35, 6, NA, 10, 39, 25, 34, 13, 22, 10, 23, 13, 25...
```

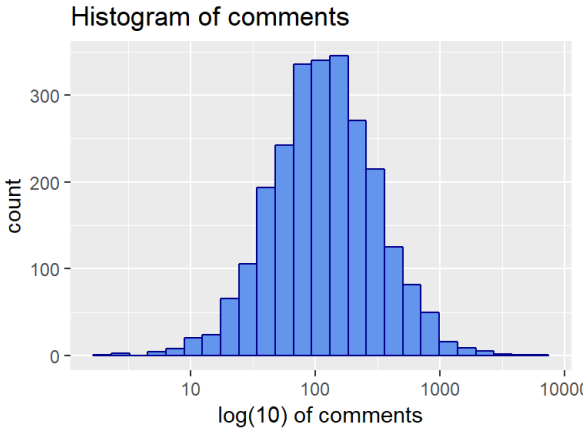
3.2. Exploratory data analysis

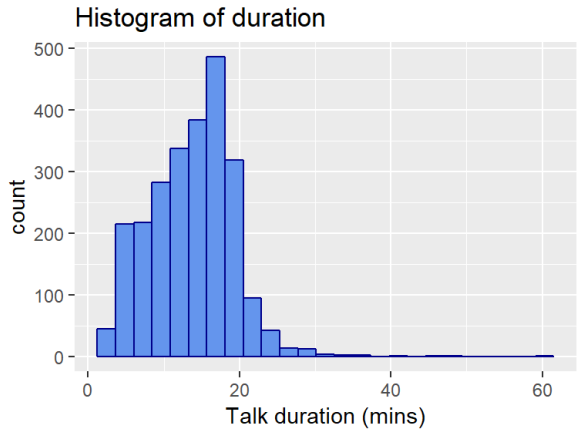
The `ted_main_data` matrix contains various numeric variables that give insight into the observation variability.

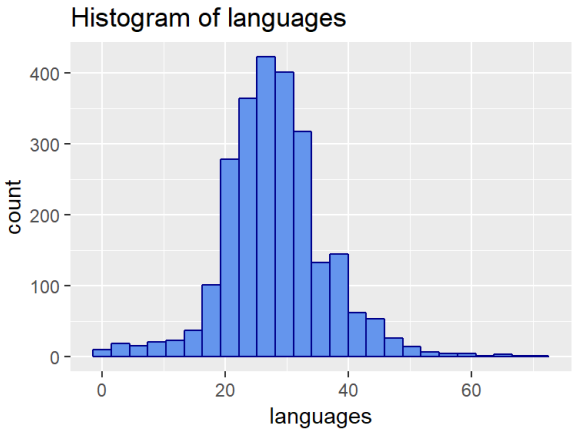
```
## Observations: 2,464
## Variables: 17
## $ comments           <int> 4553, 265, 124, 200, 593, 672, 919, 46, 852...
## $ description        <fct> "Sir Ken Robinson makes an entertaining and...
## $ duration           <int> 1164, 977, 1286, 1116, 1190, 1305, 992, 119...
## $ event              <fct> TED2006, TED2006, TED2006, TED2006, TED2006...
## $ film_date          <date> 2006-02-25, 2006-02-25, 2006-02-24, 2006-0...
## $ languages          <int> 60, 43, 26, 35, 48, 36, 31, 19, 32, 31, 27,...
## $ main_speaker       <fct> Ken Robinson, Al Gore, David Pogue, Majora ...
## $ name               <fct> "Ken Robinson: Do schools kill creativity?"...
## $ num_speaker        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ published_date     <date> 2006-06-27, 2006-06-27, 2006-06-27, 2006-0...
## $ ratings            <fct> "[{'id': 7, 'name': 'Funny', 'count': 19645...
## $ related_talks      <fct> "[{'id': 865, 'hero': 'https://pe.tedcdn.co...
## $ speaker_occupation <fct> "Author/educator", "Climate advocate", "Tec...
## $ tags               <fct> "['children', 'creativity', 'culture', 'dan...
## $ title              <fct> "Do schools kill creativity?", "Averting th...
## $ url                <fct> https://www.ted.com/talks/ken_robinson_says...
## $ views              <int> 47227110, 3200520, 1636292, 1697550, 120058...
```

A set of plots are mapped to quickly dimension various predictors (i.e., views, number of comments, talk duration, the number of languages in which the talk is available, the published date, and the number of speakers in a talk)









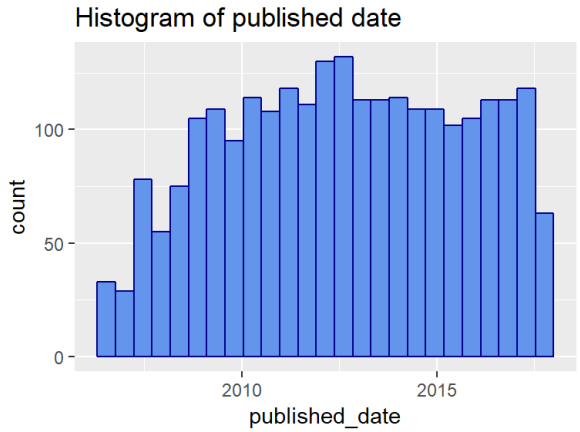
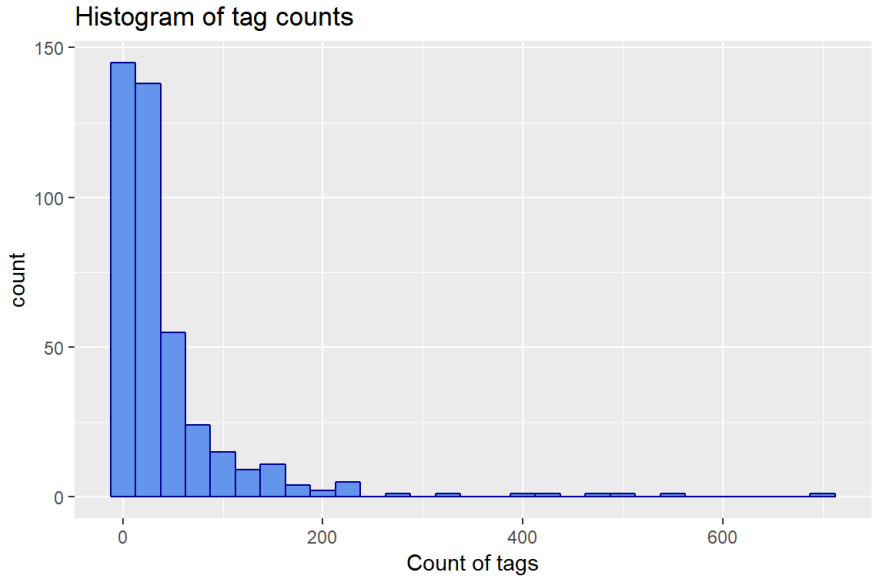


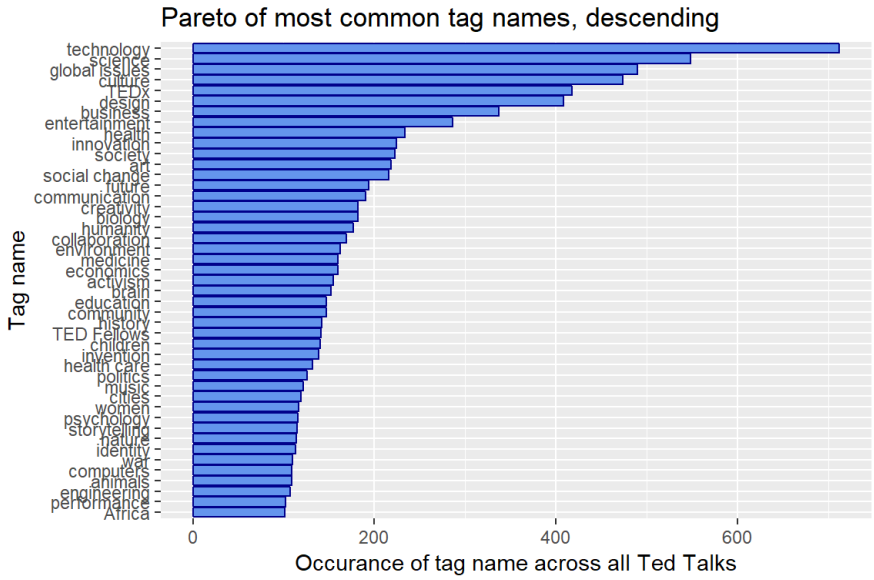
Table of number of talks given per speaker

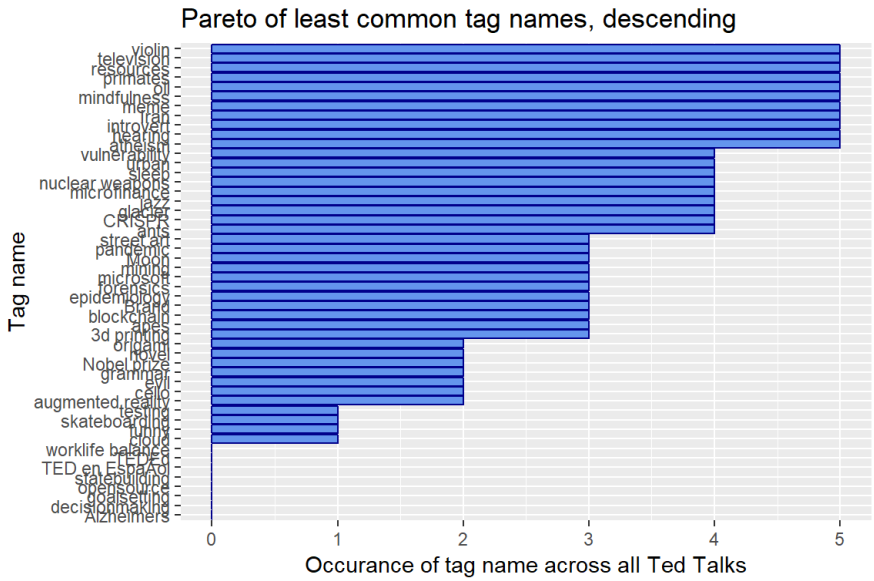
num_speaker	number_of_talks	percent_of_talks
1	2409	97.8
2	46	1.9
3	5	0.2
4	3	0.1
5	1	0.0

3.2.1. Tags

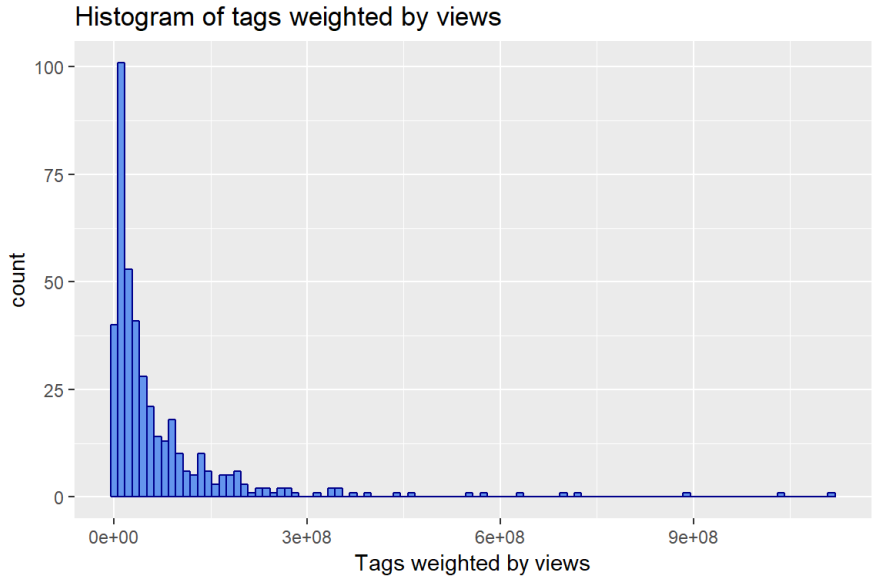
To analyse popular topics (exploring the `tags_matrix` created), we first note that 45 of the 416 total tags represent 50% of the total tags by number. The below plots shows those most common tags, and those least common for comparison.

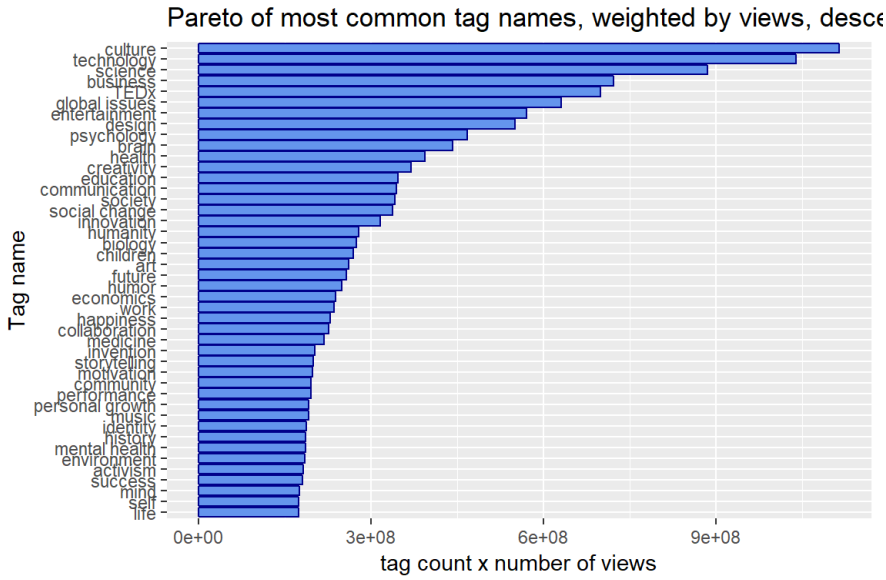


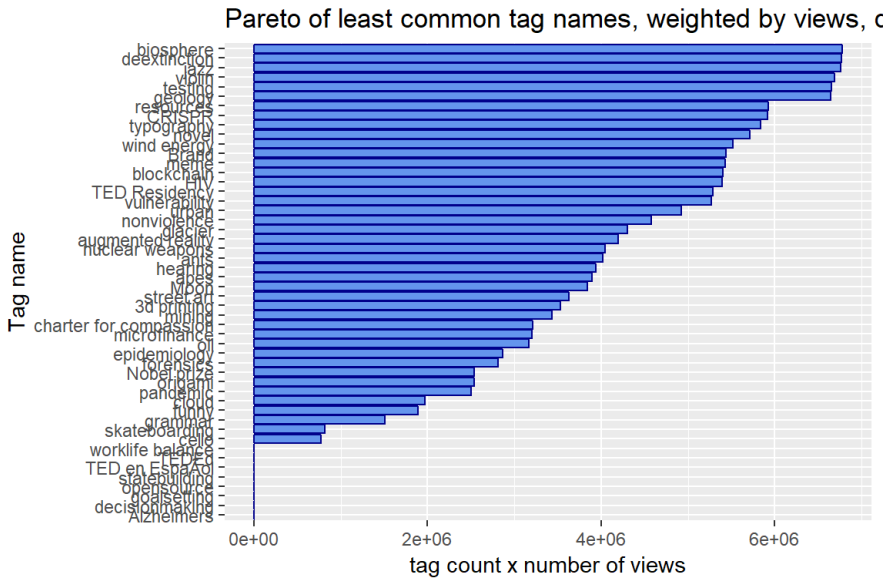




Different tags are, however, associated with talks with very different levels of viewership, thus this analysis is repeated where the tags are now weighted by the number of views of their associated talks.

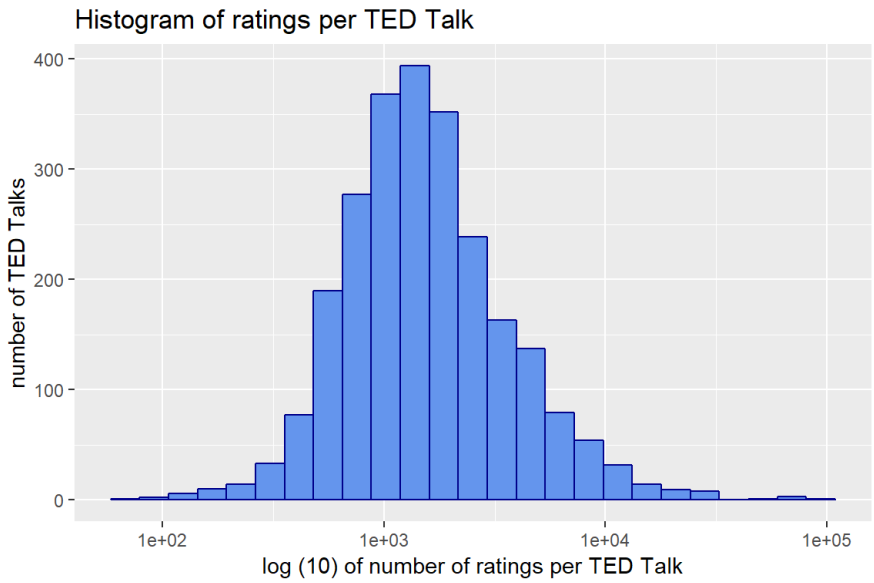






3.2.1. Ratings

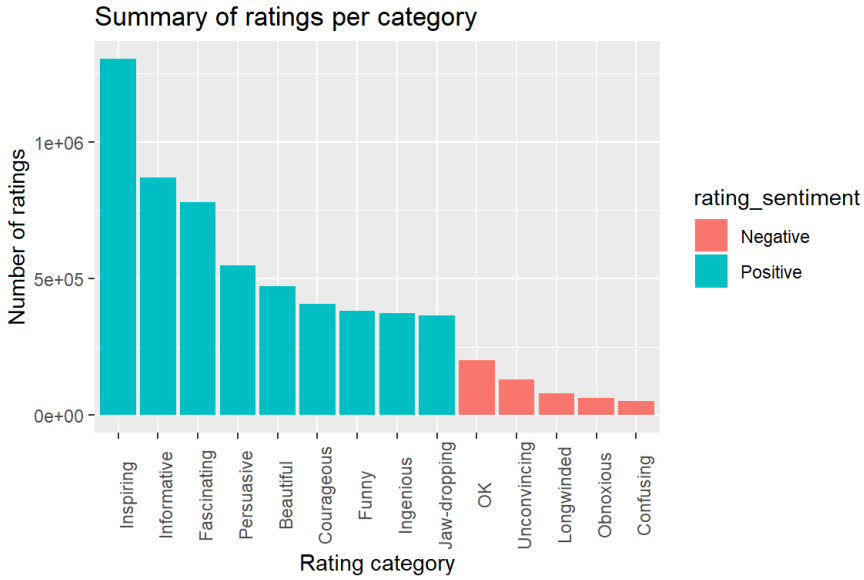
The ratings matrix across all talks records the frequency of ratings by rating type. The ratings are also broadly categorized into positive and negative ratings, as per the below code:



mean	median	sd
2446.592	1461	4225.585

```
# assigns a positive (TRUE) or negative (FALSE) flag to each of the ratings categories
ratings_sentiment <- c(TRUE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, FALSE, TRUE, TRUE, FALSE, FALSE, TRUE)
cbind(colnames(ted_main_data_ratings_matrix), ratings_sentiment) %>% kable
```

	ratings_sentiment
Funny	TRUE
Beautiful	TRUE
Ingenious	TRUE
Courageous	TRUE
Longwinded	FALSE
Confusing	FALSE
Informative	TRUE
Fascinating	TRUE
Unconvincing	FALSE
Persuasive	TRUE
Jaw-dropping	TRUE
OK	FALSE
Obnoxious	FALSE
Inspiring	TRUE

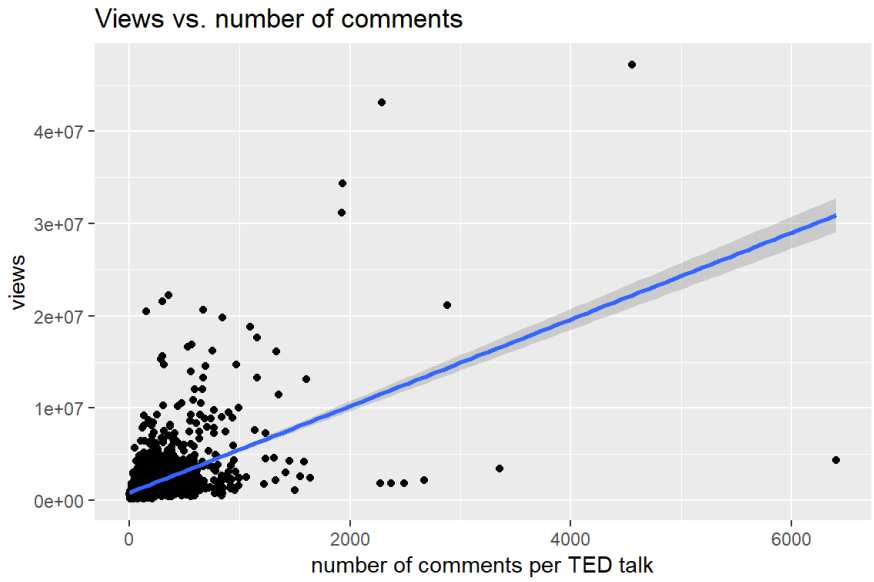


3.2.3. Correlations

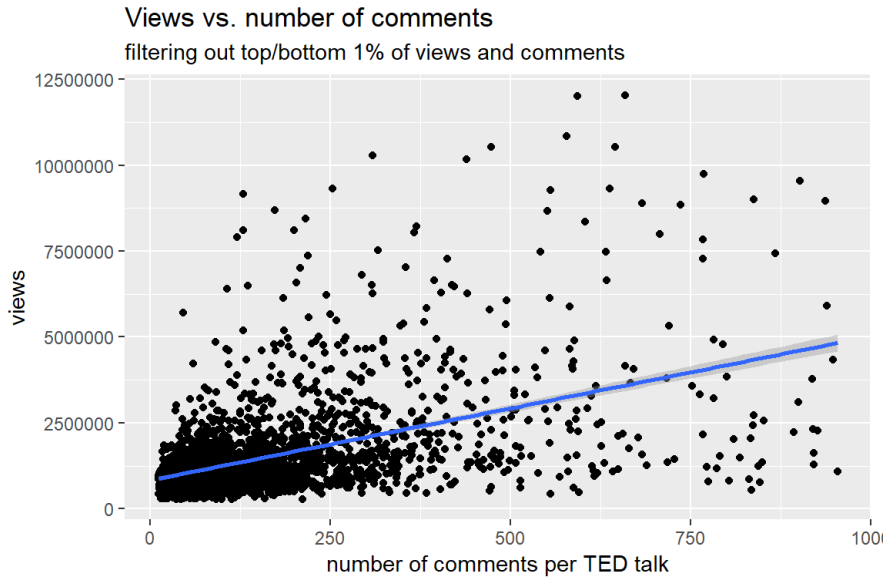
A set of parameters are selected to test correlations to `views` and hence predictive power. These various lenses on the dataset are listed below:

3.2.3.1. Views vs. number of comments per TED Talk

The hypothesis to test is that the more popular (i.e. more viewed) TED Talks will also attract a greater number of comments. We see some relationship in the below graph:



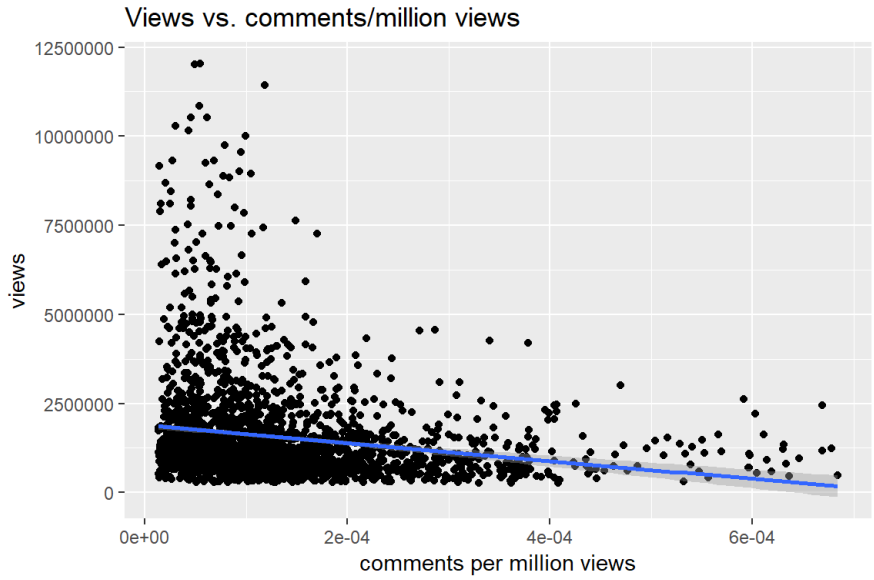
This graph shows far outliers, so the curve and data is re-run filtering out the top and bottom 1% of outliers.



The correlation of views vs. number of comments is 0.5298046

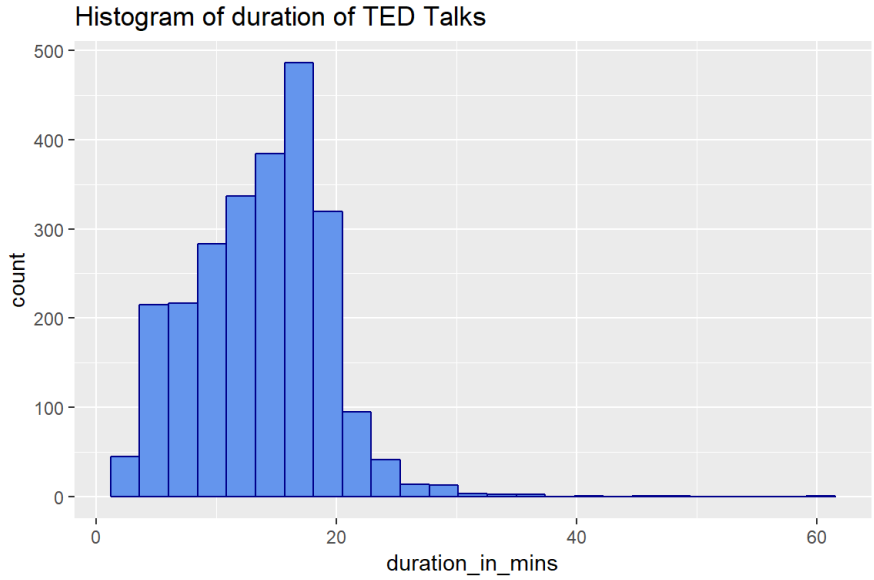
Furthermore, the ratio of comments to views is also tested, and whilst there is a moderate distribution in the comments rate, there is little predictive power in this metric (cor = -0.1292146) as show below:

```
#test "rate of comments" as % of total views, expecting that there's a stronger link btw # of comments and ratings
ted_main_data %>%
  mutate(comments_rate = comments/views) %>%
  filter(between(views, quantile(views, 0.01), quantile(views, 0.99))) %>%
  filter(between(comments_rate, quantile(comments_rate, 0.01), quantile(comments_rate, 0.99))) %>%
  ggplot(aes(x = comments_rate, y = views)) +
  geom_point() +
  geom_smooth(method = "lm", show.legend = TRUE) +
  xlab("comments per million views") +
  ggtitle("Views vs. comments/million views")
```

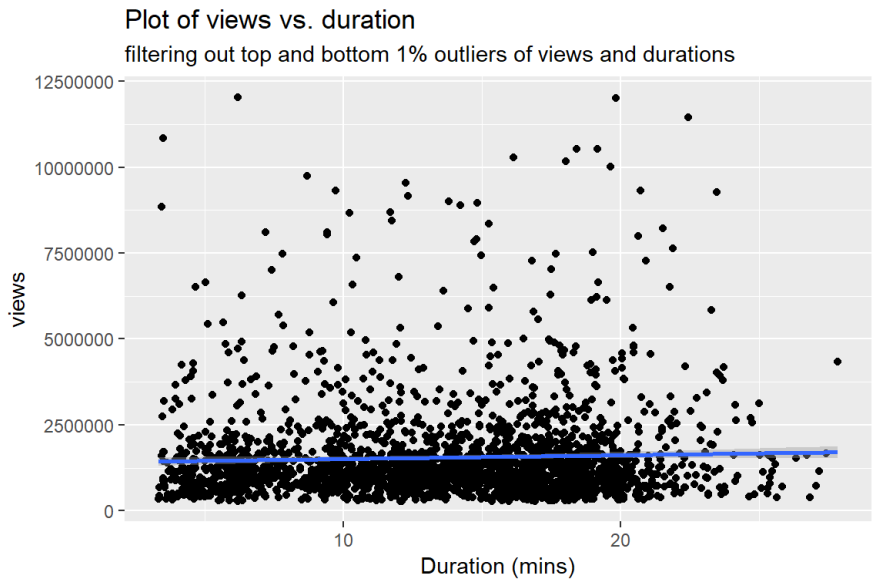


3.2.3.2. Views vs. duration of talks

The histogram of duration of talks shows a mean of 13.695 . . .



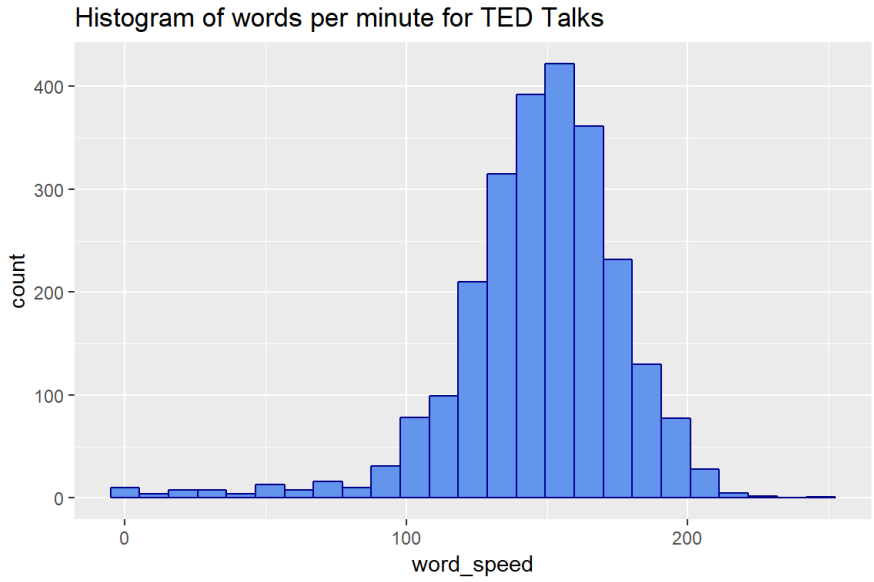
. . . and plotting views vs. duration shows weak correlation on this dimension (cor = 0.064069)



A further hypothesis is that that word count of talks and speed of delivery (words per minute of delivery) might have some moderate impact on views and popularity. This is calculated from the below code, using the `wordcount()` function from the `ngram` library.

```
#test for speed of talk - gauge of energy
library(ngram)
ted_transcripts_data_wordcount <- sapply(ted_transcripts_data$transcript, wordcount)
names(ted_transcripts_data_wordcount) <- NULL

ted_main_data %>%
  mutate(word_speed = ted_transcripts_data_wordcount/(duration/60)) %>%
  ggplot(aes(x = word_speed)) +
  geom_histogram(bins = 25, color = "blue4", fill = "cornflowerblue") +
  ggtitle("Histogram of words per minute for TED Talks")
```

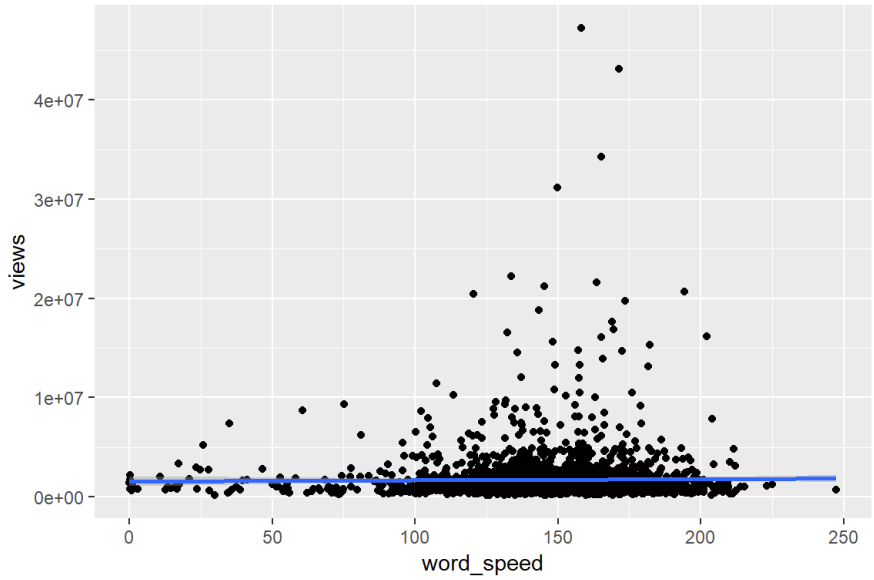



```
ted_main_data %>%
  mutate(word_speed = ted_transcripts_data_wordcount/(duration/60)) %>%
  summarize(mean_word_speed = mean(word_speed), median_word_speed = median(word_speed), sd_word_speed = sd(word_speed)) %>%
kable()
```

mean_word_speed	median_word_speed	sd_word_speed
147.1124	149.9787	29.95673

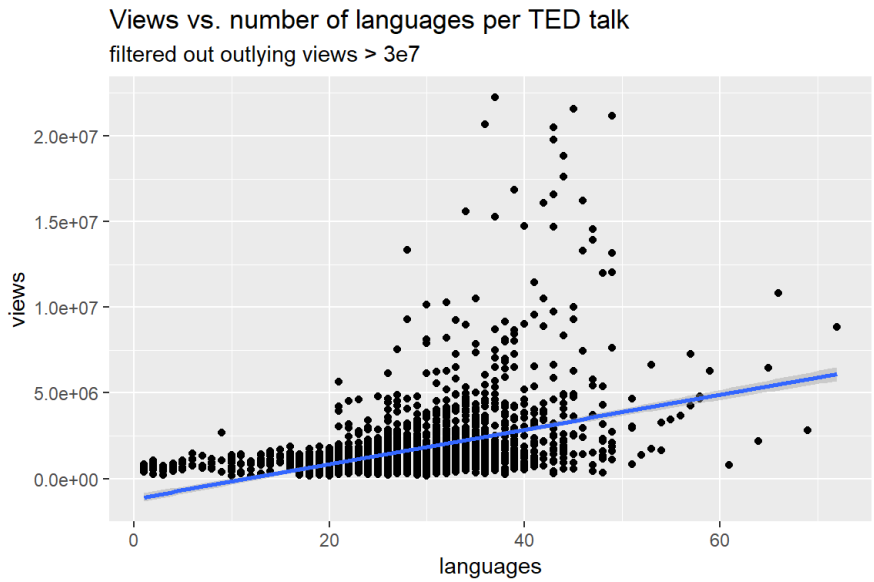
The mean word speed is 147.1, which is in line with studies of speech rates, and is visualized in the above histogram. The correlation to views, however, is again shown to be weak (cor = 0.0136656).

```
ted_main_data %>%
  mutate(word_speed = ted_transcripts_data_wordcount/(duration/60)) %>%
  ggplot(aes(x = word_speed, y = views)) +
  geom_point() +
  geom_smooth(method = "lm", show.legend = TRUE)
```



3.2.3.3. Views vs. number of languages

The number of languages that a given TED Talk is available in does show some correlation to views (cor = 0.3958925). This can be expected - as the popularity of a talk rises (i.e. `views` increase) so there is a greater probability of translation of this talk into more languages to reach a greater audience.



3.2.3.4. Views vs. number of speakers

When analyzing the number of speakers per talk (`num_speakers`) one notes a very small sample of talks with >1 speaker (only 2.23%), and illustrated in the below table. As such, this parameter is ignored for prediction purposes.

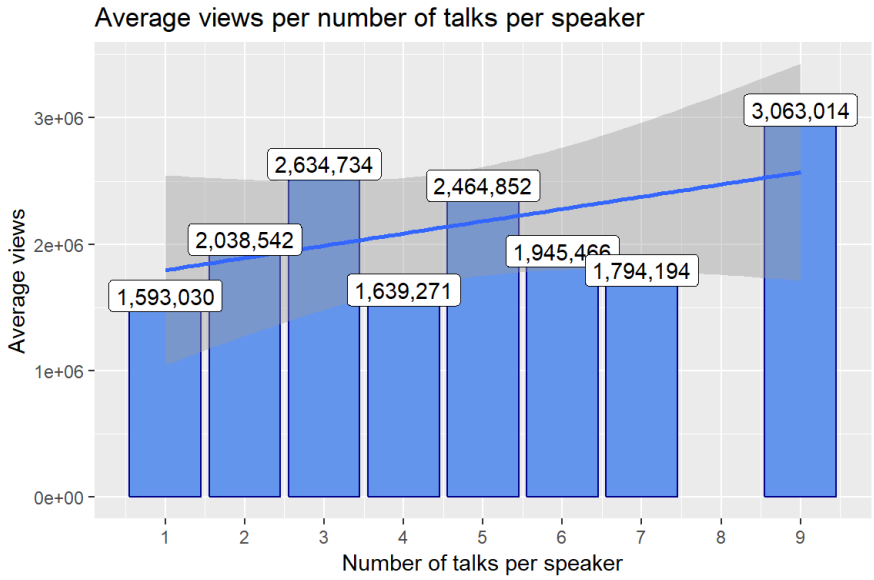
```
## # A tibble: 5 x 4
##   num_speaker     n sum_of_views average_views
##         <int> <int>        <dbl>         <dbl>
## 1           1  2409   4206580708      1746194.
## 2           2    46     65891130      1432416.
## 3           3     5      6546780      1309356
## 4           4     3      2390733       796911
## 5           5     1       182975       182975
```

3.2.3.5. Views vs. number of TED Talks per speaker

It's further hypothesized that the number of TED Talks that a given speaker has made would give some indication of their popularity, i.e. viewership.

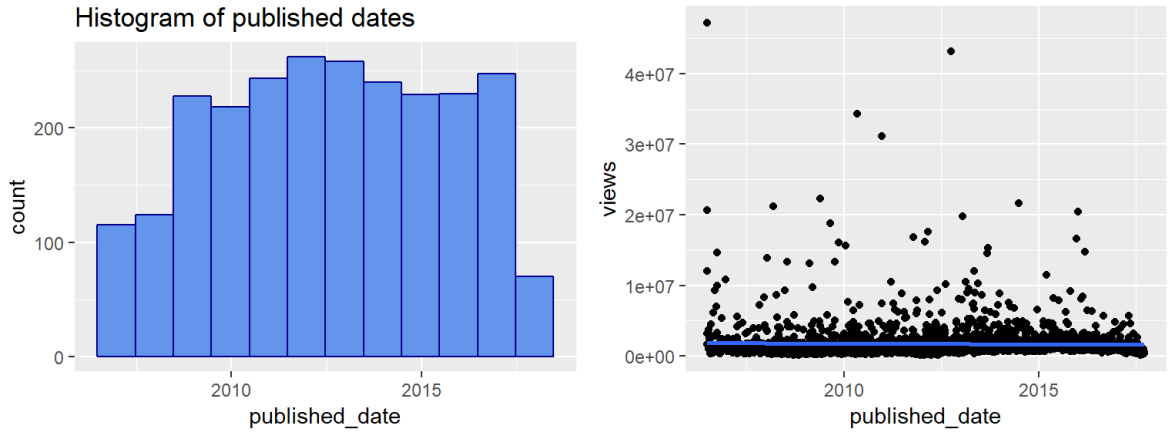
This relationship between average views vs. the number of talks given by a segment of speakers shows some directional correlation and this parameter will therefore be considered for predictive purposes.

number_of_talks_per_speaker	Number_talks_per_speaker	Total_views	Number_of_talks	Average_views
1	1	2902502111	1822	1593031
2	2	811339746	199	2038542
3	3	331976541	42	2634734
4	4	98356264	15	1639271
5	5	73945573	6	2464852
6	6	23345600	2	1945467
7	7	12559364	1	1794195
9	9	27567127	1	3063014



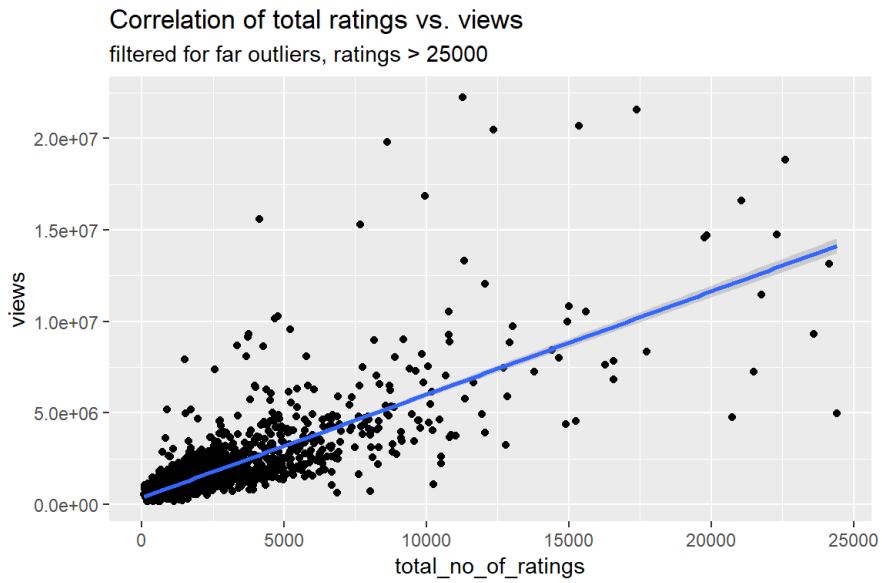
3.2.3.6. Views vs. published date

The date of publishing shows a fairly uniform distribution over time, and a low correlation (cor = -0.0245059). This parameter is also ignored in subsequent modelling.



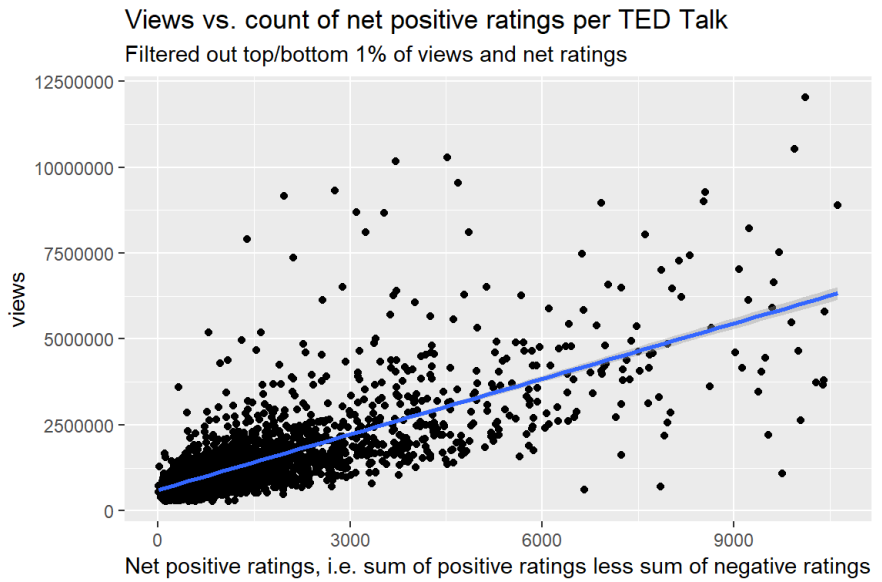
3.2.3.7. Views vs. ratings

The total number of ratings vs. views is seen to strongly correlate (cor = 0.8718445) and is illustrated in the below graph that filters out the top and bottom 1% of views and count of ratings



The correlation is further tested for the specific ratings categories, and one sees that those ratings with a more positive sentiment towards the talk show a stronger correlation to views.

	Correlation
Funny	0.7904968
Beautiful	0.7753744
Ingenious	0.7604369
Courageous	0.7271358
Longwinded	0.7048268
Confusing	0.6026254
Informative	0.5977517
Fascinating	0.5584991
Unconvincing	0.5559599
Persuasive	0.4533808
Jaw-dropping	0.4399610
OK	0.4092863
Obnoxious	0.2779880
Inspiring	0.2584371

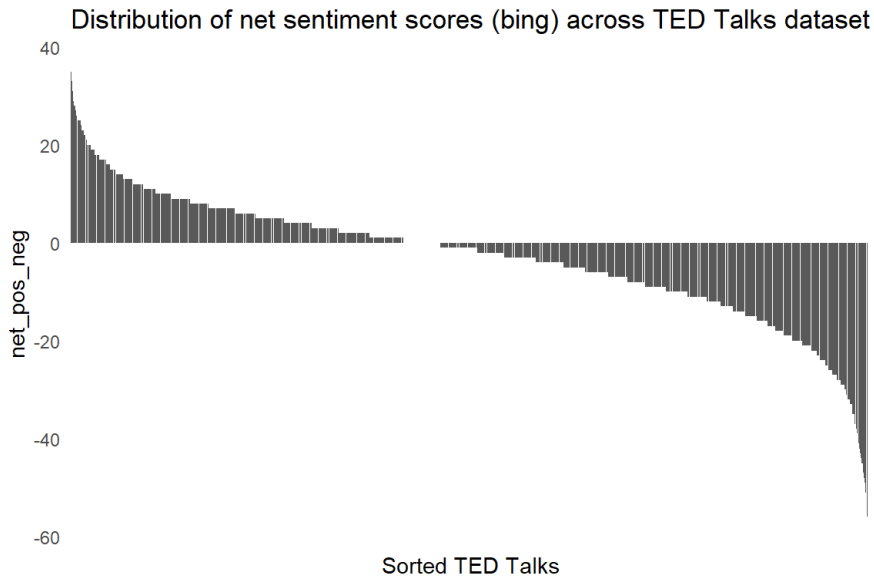


3.2.3.8. Views vs. sentiment analysis on transcripts

Finally, the dataset is tested for correlations between the number of views and the sentiment analysis applied to the associated transcripts of each TED Talk.

Bing Lexicon

Using the bing lexicon, with a count of positive and negative word sentiments per TED Talk, we can see a distribution of the net sentiment across all talks.



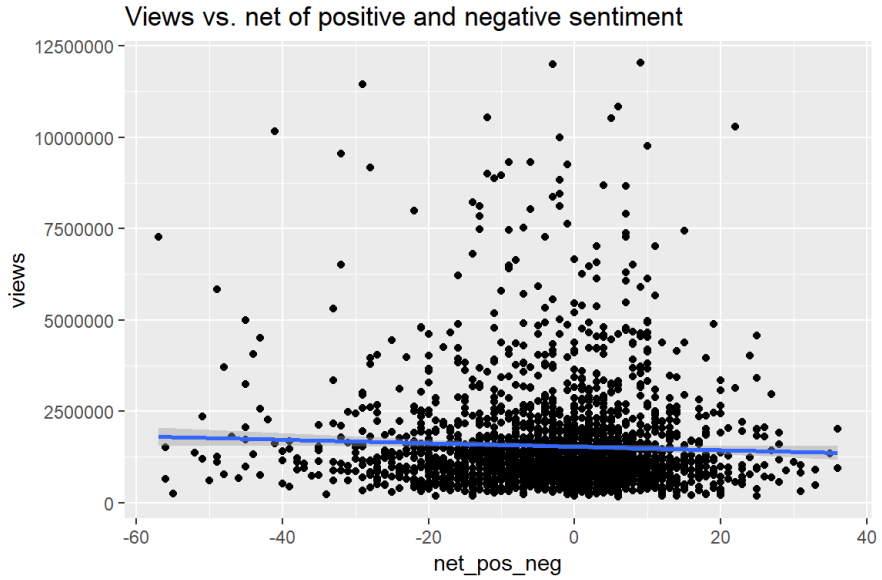
Those talks with the most positive net sentiment measured are:

title	negative	positive	net_pos_neg	views
A Darwinian theory of beauty	23	59	36	2012387
The El Sistema music revolution	17	53	36	942604
Designers – think big!	11	46	35	1340175
Everyday compassion at Google	7	40	33	893343
My wish: Find the next Einstein in Africa	31	64	33	473165
The thinking behind 50x15	17	48	31	315244
A passionate, personal case for education	9	40	31	1026589
Transition to a world without oil	26	57	31	807653
A teacher growing green in the South Bronx	24	54	30	1112964
It's time for women to run for office	21	50	29	877915

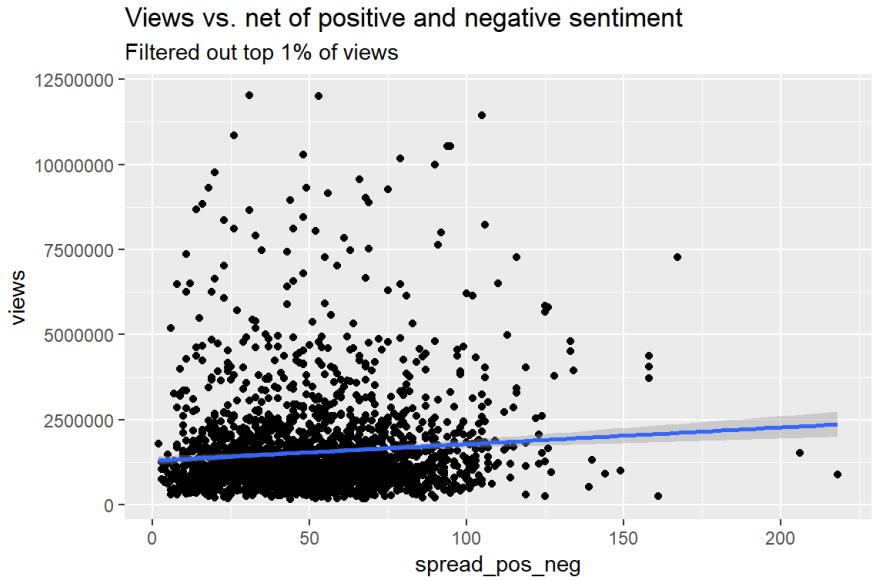
. . . and similarly the 10 most negative net sentiment talks are:

title	negative	positive	net_pos_neg	views
Why stay in Chernobyl? Because it's home.	63	14	-49	1108528
My wish: Let my photographs bear witness	87	38	-49	1262674
The psychology of evil	87	38	-49	5842377
The oil spill's unseen villains – and victims	71	21	-50	594725
What we don't know about Europe's Muslim kids	87	36	-51	1195885
My son was a Columbine shooter. This is my story	70	19	-51	2359347
The doubt essential to faith	77	25	-52	1361374
Poetry of youth and age	108	53	-55	246094
The deadly legacy of cluster bombs	72	16	-56	645572
Nationalism vs. globalism: the new political divide	131	75	-56	1514291
Depression, the secret we share	112	55	-57	7265611

Testing the correlation between net sentiment of a given talk and total views shows a correlation of -0.0116601, illustrated below:

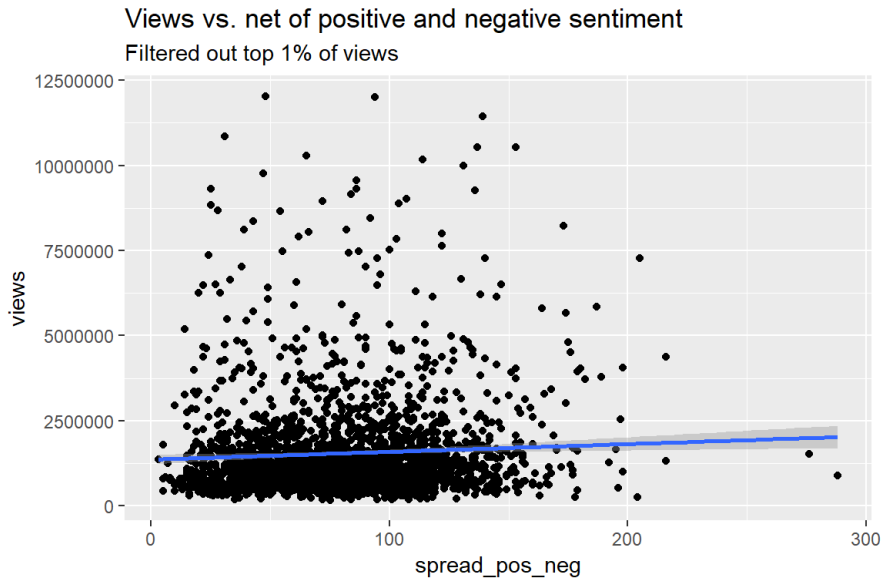


Testing correlation against the spread of sentiment, i.e. TED Talks with a wide range of positive and negative sentiment in a single talk, shows a weak correlation of 0.1002837.



NRC Emotional Lexicon

The NRC lexicon categorises single words into one of 8 categories, i.e. positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. Filtering for the positive and negative sentiment from the nrc classification, this yields very similar results to the bing lexicon, i.e. a correlation to views of 0.1002837 and a similar graphical correlation.



3.3. Prediction modelling

With a set of parameters identified to support prediction, a series of models are tested for prediction of the viewership. The relative popularity of TED Talks, as measured by viewership (`views`) is the parameter for prediction. The modelling looks to predict a talk's popularity by categorizing talks into quartiles, `views_quartile` , and measuring prediction accuracy vs. the quartile.

In practical applications, where one wants to predict the future popularity of a talk, one does not have access to viewership and ratings feedback. Some of the TED Talk dataset, e.g. languages, are likely a consequence of talk popularity (popular talks get translated into more languages), and are thus a lagging indicator of popularity.

The below models predict popularity from 2 datasets that select from parameters prepared above:

- The first is a more expansive dataset and shows higher predictive power. This includes parameters like languages and ratings. In reality, one would not have access to all of this information before a talk is published and feedback on the talk is gathered.
- The second set of models attempts to predict talk popularity based upon only those parameters known prior to the talk airing, e.g. the talk topic (`tags`), the speaker, sentiment analysis of the transcript, talk duration, the number of talks the speaker has done to date.

3.3.1. Dataset 1

We set up training dataset 1 per the below code:

```
library(caret)
set.seed(1)

#prepare final dataset for training
final_data_set <- ted_main_data %>%
  mutate(views_quartile = ntile(views, 4)) %>% # add quartiles of views as a new parameter
  mutate(number_of_ratings = rowSums(ted_main_data_ratings_matrix)) %>% #add total no. of ratings
  mutate(net_no_of_positive_ratings = rowSums(ted_main_data_ratings_matrix %*% ifelse(ratings_sentiment, 1, -1))) #add sum
 of net positive ratings

#join sentiment analysis to ted_main_data
names(ted_transcript_sentiment_bing)[-1] <- paste("bing_", names(ted_transcript_sentiment_bing)[-1], sep ="")
final_data_set <- inner_join(final_data_set, ted_transcript_sentiment_bing, "url")

names(ted_transcript_sentiment_nrc)[-1] <- paste("nrc_", names(ted_transcript_sentiment_nrc)[-1], sep = "")
final_data_set <- inner_join(final_data_set, ted_transcript_sentiment_nrc, "url")

final_data_set <- final_data_set %>%
  mutate(bing_spread_sentiment = bing_positive+bing_negative) %>% #add sentiment spread per bing
  mutate(nrc_spread_sentiment = nrc_positive+nrc_negative) %>% #add sentiment spread per nrc
  filter(complete.cases(.)) #confirm complete cases for entire dataset

# prepare training and test sets
training_sample_percent <- 1 # choose to first model off of a subset of ted_main_data (to prove algorithms and increase run
speed whilst testing models)
final_data_subset <- final_data_set[sample(final_data_set$comments, size = floor(nrow(final_data_set) * training_sample_per
cent), replace = FALSE), ]
final_data_subset <- final_data_subset %>% filter(complete.cases(.))

train_index <- createDataPartition(final_data_subset$views, times = 1, p = 0.8, list = FALSE)
train_set <- final_data_subset[train_index, ]
test_set <- final_data_subset[-train_index, ]
```

The 3 models that follow - `knn` , `rborist` and `ranger` - are trained on the following parameters, selected from the compiled training set:

- `comments`
- Talk `duration`
- `languages`
- The main speaker, `main_speaker`
- The number of talks by the speaker, `num_speaker`
- The total `number_of_ratings` received per talk
- The nett number of positive ratings received, `net_no_of_positive_ratings`
- The spread of sentiment using the bing lexicon, i.e. the count of positive sentiments + the count of negative sentiments - `bing_spread_sentiment`
- The spread of sentiment using the NRC lexicon - `nrc_spread_sentiment`

The first round of models do not include:

- The topic tags as a predictor (given the exponential growth in the modelling parameters when including these)
- The more detailed categorisations of ratings
- The specific sentiments isolated by the NRC sentiment analysis (beyond simply the positive and negative impacts)
- Other parameters which showed low correlation from the exploratory analysis phase, e.g. published_date

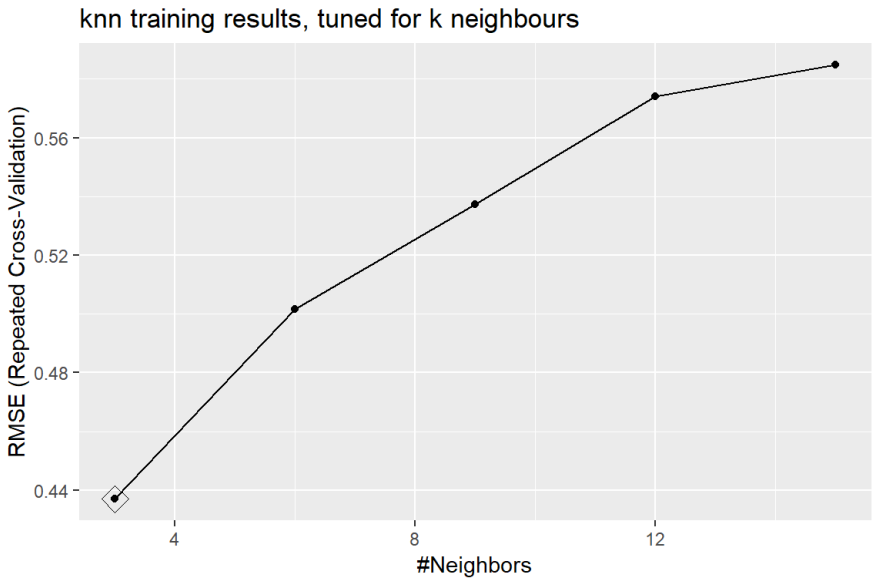
Model 1: knn

A k-nearest neighbours model is set up with 5-fold repeated cross-validation (3 repeats) and a tuning range of k neighbours of `seq(3, 15, 3)` .

```
# model 1: knn
ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 3, verboseIter = FALSE)
k_tune_df <- data.frame(k = seq(3, 15, 3))
train_knn <- train(form = views_quartile ~ .,
                   data = train_set %>%
                     select(views_quartile, comments, duration, languages,
                            num_speaker, number_of_ratings,
                            net_no_of_positive_ratings, bing_spread_sentiment, nrc_spread_sentiment),
                   method = "knn",
                   trControl = ctrl,
                   preProcess = c("center", "scale"),
                   tuneGrid = k_tune_df)

conf_matrix_knn <- confusionMatrix(data = as.factor(round(predict(train_knn, test_set),0)), reference = as.factor(test_set
$views_quartile))
```

This model yields an overall accuracy of 0.8940678, and the tuned model is optimized at k = 3, seen in the below plot.



The table of predicted vs. reference results furthermore shows that the error rate for those talks which are classified as more than 1 quartile removed from actual is very low.

```
conf_matrix_knn$table
```

```
##            Reference
## Prediction   1   2   3   4
##          1 170  10   1   2
##          2   5  77   9   3
##          3   2   2  69  12
##          4   0   1   3 106
```

A simple function is defined to calculate the accuracy to within 1 quartile, `accuracy_1_quartile` , listed below:

```
accuracy_1_quartile <- function(x) {
  1-sum(x[1, 3:4], x[2, 4], x[3, 1], x[4, 1:2])/sum(x)
}
```

. . . and calling `accuracy_1_quartile` for the results table from the knn fit yields an accuracy of 0.98.

The specific test results for each of the quartiles are also displayed below:

```
train_knn$results %>% kable
```

k	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
3	0.4369947	0.8743079	0.1489690	0.0523352	0.0286568	0.0186724
6	0.5015310	0.8343605	0.2338093	0.0505587	0.0324795	0.0283907
9	0.5373619	0.8088714	0.3064087	0.0522242	0.0359495	0.0330512
12	0.5739695	0.7806808	0.3634930	0.0487253	0.0370973	0.0319213

k	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
15	0.5848671	0.7715575	0.3902046	0.0422890	0.0340376	0.0284256

Model 2: Rborist

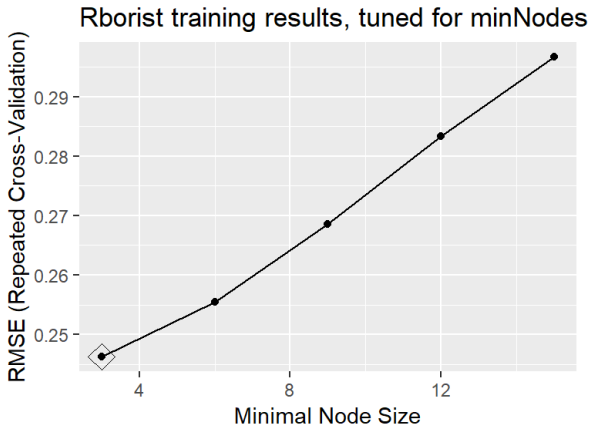
The Rborist training method implements a random forest algorithm. This is set up with 5-fold repeated cross validation for 3 repeats, and is tuned for the minimal number of rows to split a node (`minNode = seq(3,15,3)`). This model is shown below.

```
# model 2: rborist
ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 3, verboseIter = FALSE)
tune_Rborist <- data.frame(predFixed = 2, minNode = seq(3,15,3))
train_Rborist <- train(form = views_quartile ~ .,
                       data = train_set %>%
                         select(views_quartile, comments, duration, languages,
                                num_speaker, number_of_ratings,
                                net_no_of_positive_ratings, bing_spread_sentiment, nrc_spread_sentiment),
                       method = "Rborist",
                       trControl = ctrl,
                       preProcess = c("center", "scale"),
                       tuneGrid = tune_Rborist)

conf_matrix_rborist <- confusionMatrix(data = as.factor(round(predict(train_Rborist, test_set),0)), reference = as.factor(t
est_set$views_quartile))
```

The model optimizes for `minNode` = 3 and delivers an overall accuracy of 0.9576271. The model results are included below:

```
ggplot(train_Rborist, highlight = TRUE) + ggtitle("Rborist training results, tuned for minNodes")
```




```
conf_matrix_rborist$table
```

```
##          Reference
## Prediction   1   2   3   4
##          1 175   3   2   1
##          2   2  86   2   1
##          3   0   1  76   6
##          4   0   0   2 115
```

```
conf_matrix_rborist$byClass %>% kable
```

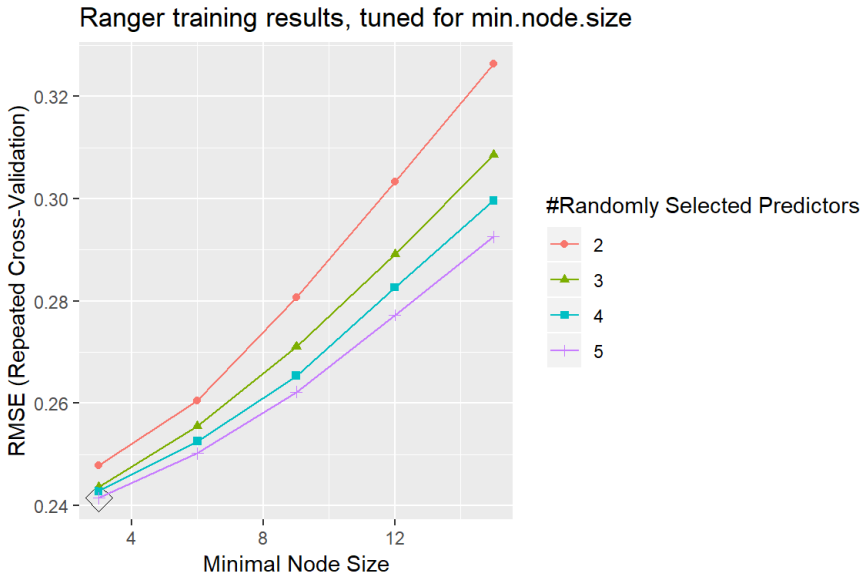
	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: 1	0.9887006	0.9796610	0.9668508	0.9931271	0.9668508	0.9887006	0.9776536	0.3750000	0.3707627	0.3834746	0.9841808
Class: 2	0.9555556	0.9869110	0.9450549	0.9895013	0.9450549	0.9555556	0.9502762	0.1906780	0.1822034	0.1927966	0.9712333
Class: 3	0.9268293	0.9820513	0.9156627	0.9845758	0.9156627	0.9268293	0.9212121	0.1737288	0.1610169	0.1758475	0.9544403
Class: 4	0.9349593	0.9942693	0.9829060	0.9774648	0.9829060	0.9349593	0.9583333	0.2605932	0.2436441	0.2478814	0.9646143

Finally, the accuracy to within one quartile (`accuracy_1_quartile`) is also calculated as 0.99.

Model 3: Ranger

The `ranger` method (RANdom forest GEneRator) is cited as a random forest training method well-suited to higher dimensionality data and yields fast and memory-efficient results. The `ranger` model is also set up with 5-fold repeated cross validation for 5 repeats. Ranger optimizes on `min.node.size` per the below code:

The ranger model delivers an overall accuracy of 0.9639831, and an accuracy to within 1 quartile of 0.9894068. The model results are further visualised below:




```
##          Reference
## Prediction   1   2   3   4
##          1 175   1   2   1
##          2   2  88   2   2
##          3   0   1  76   4
##          4   0   0   2 116
```

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: 1	0.9887006	0.9864407	0.9776536	0.9931741	0.9776536	0.9887006	0.9831461	0.3750000	0.3707627	0.3792373	0.9875706
Class: 2	0.9777778	0.9842932	0.9361702	0.9947090	0.9361702	0.9777778	0.9565217	0.1906780	0.1864407	0.1991525	0.9810355
Class: 3	0.9268293	0.9871795	0.9382716	0.9846547	0.9382716	0.9268293	0.9325153	0.1737288	0.1610169	0.1716102	0.9570044
Class: 4	0.9430894	0.9942693	0.9830508	0.9802260	0.9830508	0.9430894	0.9626556	0.2605932	0.2457627	0.2500000	0.9686794

3.3.2. Dataset 2

At time of publication of a TED Talk, not all information will be available, e.g. comments, ratings, translation into further languages, etc. Further models are trained on the following parameters to profile the topic and speaker popularity:

- Talk `duration`
- The `main_speaker`
- The number of talks by the speaker, `num_speaker`
- The spread of sentiment using the bing lexicon, i.e. the count of positive sentiments + the count of negative sentiments - `bing_spread_sentiment`
- The spread of sentiment using the NRC lexicon - `nrc_spread_sentiment`
- The topic `tags`

The second dataset is prepared using the following code:

```
#Dataset 2: no a priori parameters to train on
set.seed(1)
#prepare final dataset for training
final_data_set <- ted_main_data %>%
  mutate(views_quartile = ntile(views, 4)) # add quartiles of views as a new parameter

colnames(tags_matrix) <- paste("tag_", colnames(tags_matrix), sep = "") # rename columns in tags matrix
final_data_set <- cbind(final_data_set, tags_matrix) # add on tags matrix to ted_main_data

#join sentiment analysis to ted_main_data
final_data_set <- inner_join(final_data_set, ted_transcript_sentiment_bing, "url")

#names(ted_transcript_sentiment_nrc)[-1] <- paste("nrc_", names(ted_transcript_sentiment_nrc)[-1], sep = "")
final_data_set <- inner_join(final_data_set, ted_transcript_sentiment_nrc, "url")

final_data_set <- final_data_set %>%
  mutate(bing_spread_sentiment = bing_positive+bing_negative) %>% #add sentiment spread per bing
  mutate(nrc_spread_sentiment = nrc_positive+nrc_negative) %>% #add sentiment spread per bing
  filter(complete.cases(.)) #confirm complete cases for entire dataset

#filter out parameters not used for training, incl. those that will only be known after talk release and feedback from view
ers
final_data_set <- final_data_set %>% select(-languages, -ratings, -description, -related_talks, -event, -name, -tags,
                                            -speaker_occupation, -main_speaker, -title, -url, -views)

# prepare training and test sets
training_sample_percent <- 1 # choose to first model off of a subset of ted_main_data (to prove algorithms and increase run
speed whilst testing models)
final_data_subset <- final_data_set[sample(final_data_set$comments, size = floor(nrow(final_data_set) * training_sample_per
cent), replace = FALSE), ]
final_data_subset <- final_data_subset %>% filter(complete.cases(.))

train_index <- createDataPartition(final_data_subset$views, times = 1, p = 0.8, list = FALSE)
train_set <- final_data_subset[train_index, ]
test_set <- final_data_subset[-train_index, ]
```

This second dataset has many more parameters, 1897 rows x 436 colums, and as such a single training method, `ranger`, is applied to the dataset. `Ranger` is well-suited to fast random forest implementations of higher dimensional datasets. In earlier code development it was seen to perform better than `knn`, `rborist` and simple `rf` methods.

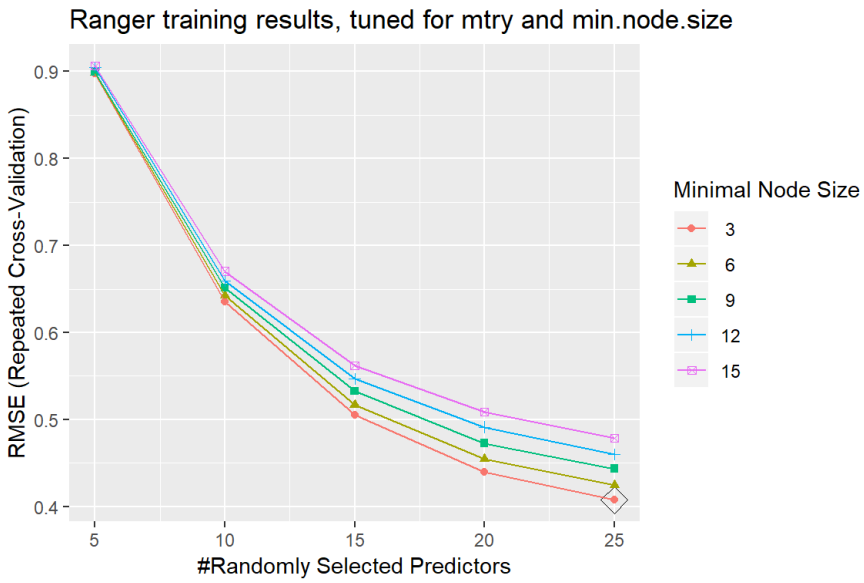
Model 1: Ranger

The `ranger` model on dataset 2 is configured and executed as per below, with a 5-fold 3 repeat cross validation and tuning for a range of `mtry` and `min.node.size`:

```
# model 1: ranger
ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 3, verboseIter = FALSE)
tune_ranger <- expand.grid(.mtry = seq(5,25,5), .splitrule = "extratrees", .min.node.size = seq(3,15,3))

train_ranger2 <- train(form = views_quartile ~ .,
                   data = train_set,
                   method = "ranger",
                   trControl = ctrl,
                   preProcess = c("center", "scale"),
                   tuneGrid = tune_ranger)
conf_matrix_ranger2 <- confusionMatrix(data = as.factor(round(predict(train_ranger2, test_set),0)),
                                   reference = as.factor(test_set$views_quartile))
```

The ranger model delivers an overall accuracy of 0.9217759, and an accuracy to within 1 quartile of 0.9852008. The model results are further visualised below:




```
##          Reference
## Prediction   1   2   3   4
##          1 150   0   1   0
##          2  16 100   3   4
##          3   2   1  67  10
##          4   0   0   0 119
```

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: 1	0.8928571	0.9967213	0.9933775	0.9440994	0.9933775	0.8928571	0.9404389	0.3551797	0.3171247	0.3192389	0.9447892
Class: 2	0.9900990	0.9381720	0.8130081	0.9971429	0.8130081	0.9900990	0.8928571	0.2135307	0.2114165	0.2600423	0.9641355
Class: 3	0.9436620	0.9676617	0.8375000	0.9898219	0.8375000	0.9436620	0.8874172	0.1501057	0.1416490	0.1691332	0.9556618
Class: 4	0.8947368	1.0000000	1.0000000	0.9604520	1.0000000	0.8947368	0.9444444	0.2811839	0.2515856	0.2515856	0.9473684

4. Results

4.1. Results - parameters with audience feedback data

Consolidating these results, and evaluating on overall average, we see the relative performance of the various models on the given train and test datasets.

```
df_dataset1_results <- data.frame(name = character(),
                                  accuracy = numeric())

df_dataset1_results <- rbind(df_dataset1_results,
                          data_frame(name = "knn", accuracy = conf_matrix_knn$overall[["Accuracy"]]),
                          data_frame(name = "rborist", accuracy = conf_matrix_rborist$overall[["Accuracy"]]),
                          data_frame(name = "ranger", accuracy = conf_matrix_ranger$overall[["Accuracy"]]))

tmp <- c(accuracy_1_quartile(conf_matrix_knn$table),
         accuracy_1_quartile(conf_matrix_rborist$table),
         accuracy_1_quartile(conf_matrix_ranger$table))

df_dataset1_results <- cbind(df_dataset1_results, tmp)
names(df_dataset1_results)[3] <- "Accuracy to 1 quartile"
df_dataset1_results %>% kable
```

name	accuracy	Accuracy to 1 quartile
knn	0.8940678	0.9809322
rborist	0.9576271	0.9915254

name	accuracy	Accuracy to 1 quartile
ranger	0.9639831	0.9894068

From the first dataset we see that the ranger model performs best, with an overall accuracy of 0.9639831 and accuracy to within 1 quartile of 0.9894068.

For the purposes of selecting media content likely to be popular to audiences, this accuracy is very high, and there will also be a step of human oversight before finalizing programming to validate model recommendations.

4.2. Results - parameters at date of publication

Predicting talk popularity at time of publication is more difficult, given lack of audience feedback, hence the training parameters were significantly broadened (e.g. `tags`, `detailed sentiment parameters`). From the second dataset, the ranger model selected performs well (given the train and test sets under consideration), with an overall accuracy of 0.9217759, and an accuracy to within 1 quartile of 0.9852008.

This accuracy is also sufficiently high to validate the model use as a decision-making tool for media programming.

5. Conclusions and next steps

This project explores methods to predict TED Talk popularity. One notes that the ability to predict talk popularity (as measured by viewership levels) is strengthened with feedback information from the talks after publication, e.g. with comments/ratings/languages data available. The overall accuracy of classification of TED Talks into quartiles of viewership is 0.964 using data parameters gathered after a talk has been published and feedback is received.

The accuracy drops to 0.9218 (and 0.9852 if measuring accuracy to within one quartile of actual classification).

This indicates that prediction of viewership from the more limited parameters of speaker, topic tags and transcript sentiment analysis is still possible and of value in commercial application. To further improve upon this prediction, further investigation is proposed to:

- Refine the tuning parameters on the models applied in this project;
- Test alternative goal functions to evaluate overall model strength, e.g. forecasting actual viewership, F1 scores to incorporate precision and recall, etc.;
- Conduct principle component analysis to identify greatest sources of variance driving model prediction, and use this information to deepen the datasets for relevant predictors;
- Evaluate additional training algorithms beyond the knn, random forest and simple neural network approaches above;
- Augment the existing TED Talks dataset with other media content datasets, to broaden the sample size to predict content popularity;
- Augment with audience data to better profile popularity to the target audience.

I would like to acknowledge and thank Prof Irizarry and his broader team for creating and investing time into this HarvardX Data Science course, without which I would not have been able to learn about and explore this topic to this extent.