

# 632 Project

```
library(tidyverse)
library(dplyr)
library(stringr)
library(randomForest)
library(vip)
library(rpart)
library(rpart.plot)
library(caret)
library(tidytext)
library(tidyr)
library(MASS)
library(car)
```

## Research Data

The dataset contains **2515 unique videos** and their **subtitles** from over **91 different YouTubers**, ranging from all different kinds of categories.

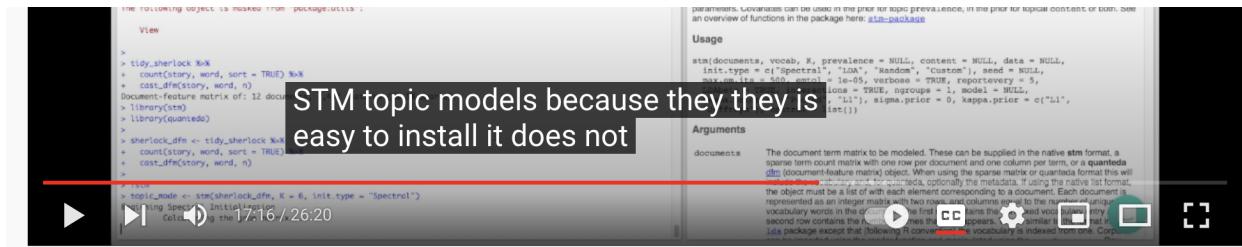
```
df_raw <- read.csv("data.csv")
head(df_raw)

##           Id      Channel   Subscribers
## 1 FozCkl1xj-w     JRE Clips 6.28M subscribers
## 2 RN8yoi-e2yc    Mythical Kitchen 1.9M subscribers
## 3 IugcIAAZJ2M    Munchies 4.59M subscribers
## 4 JiE06F8i0eU Parks and Recreation 282K subscribers
## 5 1T4XMNN4bNM       Vsauce 17.4M subscribers
## 6 OZWGeidvrJw    Doctor Who 1.59M subscribers
##                                     Title CC
## 1           Former CIA Agent Breaks Down Jeffrey Epstein Case  0
## 2 $420 Pizza Hut Stuffed Crust Pizza | Fancy Fast Food | Mythical Kitchen  1
## 3           The Iconic $1 Pizza Slice of NYC | Street Food Icons  0
## 4           Ron Swanson: The Papa of Pawnee | Parks and Recreation  0
## 5           What's The Most Dangerous Place on Earth?  1
## 6 The Doctor Defeats the Abzorbaloff | Love and Monsters | Doctor Who  1
##          URL   Released   Views
## 1 https://www.youtube.com/watch?v=FozCkl1xj-w 2 years ago 7.9M views
## 2 https://www.youtube.com/watch?v=RN8yoi-e2yc           2.7M views
## 3 https://www.youtube.com/watch?v=IugcIAAZJ2M 2 years ago 11M views
## 4 https://www.youtube.com/watch?v=JiE06F8i0eU 3 years ago 2.3M views
## 5 https://www.youtube.com/watch?v=1T4XMNN4bNM 9 years ago 21M views
## 6 https://www.youtube.com/watch?v=OZWGeidvrJw 7 years ago 8.5M views
##          Category
## 1         Blog
```

```

## 2          Food
## 3          Food
## 4 Entertainment,Comedy
## 5          Science
## 6      Entertainment
##
## 1
## 2 - Oh, that's dirty.\n- Wow! - Whoa.\n- You're a dirty girl. (upbeat music) - Hey man. - What'd you
## 3
## 4
## 5
## 6
##   Length
## 1 13:32
## 2 24:26
## 3 7:51
## 4 10:06
## 5 9:29
## 6 4:20

```



### Topic modeling with R and tidy data principles

48,633 views • Dec 18, 2017

912 DISLIKE SHARE DOWNLOAD CLIP SAVE ...

## Research Goal

Do a sentiment analysis on the subtitles and find the best multiple linear regression model to predict the number of views using Subscribers, CC, Released, Category, Sentiment and Length.

## Tasks

1. Data Cleaning.
2. Conduct a sentiment analysis on the subtitles.
3. Try various statistical models like linear regression, decision tree and random forest.
4. Compare these models in terms of prediction performance and interpretability.

## Data Cleaning

```
df_raw %>% dplyr::select(URL, Channel, Views, Subscribers, Released, Length) %>% head(10)
```

	URL	Channel	Views
## 1	<a href="https://www.youtube.com/watch?v=FozCkl1xj-w">https://www.youtube.com/watch?v=FozCkl1xj-w</a>	JRE Clips	7.9M views
## 2	<a href="https://www.youtube.com/watch?v=RN8yoi-e2yc">https://www.youtube.com/watch?v=RN8yoi-e2yc</a>	Mythical Kitchen	2.7M views
## 3	<a href="https://www.youtube.com/watch?v=IugcIAAZJ2M">https://www.youtube.com/watch?v=IugcIAAZJ2M</a>	Munchies	11M views
## 4	<a href="https://www.youtube.com/watch?v=JiE06F8i0eU">https://www.youtube.com/watch?v=JiE06F8i0eU</a>	Parks and Recreation	2.3M views
## 5	<a href="https://www.youtube.com/watch?v=1T4XMNN4bNM">https://www.youtube.com/watch?v=1T4XMNN4bNM</a>	Vsauce	21M views
## 6	<a href="https://www.youtube.com/watch?v=OZWGeidvrJw">https://www.youtube.com/watch?v=OZWGeidvrJw</a>	Doctor Who	8.5M views
## 7	<a href="https://www.youtube.com/watch?v=YiEj9mrqTNO">https://www.youtube.com/watch?v=YiEj9mrqTNO</a>	A&E	14M views
## 8	<a href="https://www.youtube.com/watch?v=PZFLM2DVQHs">https://www.youtube.com/watch?v=PZFLM2DVQHs</a>	EpicNameBro	502K views
## 9	<a href="https://www.youtube.com/watch?v=CoDpjzPzAh0">https://www.youtube.com/watch?v=CoDpjzPzAh0</a>	Insider News	583K views
## 10	<a href="https://www.youtube.com/watch?v=VT128ElBWkM">https://www.youtube.com/watch?v=VT128ElBWkM</a>	Incognito Mode	2.2M views
##	Subscribers      Released      Length		
## 1	6.28M subscribers	2 years ago	13:32
## 2	1.9M subscribers		24:26
## 3	4.59M subscribers	2 years ago	7:51
## 4	282K subscribers	3 years ago	10:06
## 5	17.4M subscribers	9 years ago	9:29
## 6	1.59M subscribers	7 years ago	4:20
## 7	7.93M subscribers	2 years ago	20:54
## 8	389K subscribers	6 years ago	33:13
## 9	216K subscribers	6 months ago	7:17
## 10	1.62M subscribers	4 years ago	4:20

Looking at the data, we notice several problems in the data, like:

1. Views: The Views variable is in string format and the units are different, like “10K views”, “10M views”. We prefer it to be in number and in the same unit in order to conduct statistical analysis.
2. Subscribers: The same problems as Views. The Subscribers variable is like “10K subscribers”, “10M subscribers”.
3. Length: The video length is in string format, like “12:00”, “1:12:00”. We need it to be in number and in the same unit.
4. Released: The Released variable is in string format, like “2 years ago”, “10 month ago”. We need it to be in number and in the same unit.

Therefore, we need to do data cleaning first.

```
# Unify units and convert string to number, like: 10K views -> 10, 10M views -> 10000
cleanViews <- function(str) {
  str <- str_remove(str, " views")
  last <- str_sub(str, -1)
  views <- str %>% str_remove(last) %>% as.numeric()
  if (last == "M") return(1000*views)
  else return(views)
}

# Unify units and convert string to number, like: 10K subscribers -> 10, 10M subscribers -> 10000
cleanSubscribers <- function(str) {
  str <- str_remove(str, " subscribers")
```

```

last <- str_sub(str, -1)
views <- str %>% str_remove(last) %>% as.numeric()
if (last == "M") return(1000*views)
else return(views)
}

# Convert time in string format to number of minutes, like: 12:00 -> 12, 1:12:00 -> 72
cleanLength <- function(str) {
  list <- str_split(str, ":")
  len <- length(list[[1]])
  if (len == 3) {
    h <- as.numeric(list[[1]][1])
    m <- as.numeric(list[[1]][2])
    return((m + 1) + 60*h)
  } else {
    m <- as.numeric(list[[1]][1])
    return(m+1)
  }
}

# Convert time to number of months ago, like: 1 years ago -> 12, 10 months ago to 10
cleanReleased <- function(str) {
  str <- str_remove(str, "Streamed ")
  list <- str_split(str, " ")
  if (list[[1]][2] == "years") return(as.numeric(list[[1]][1])*12)
  else return(as.numeric(list[[1]][1]))
}

# Remove NAs
df <- df_raw %>%
  na.omit() %>%
  filter(
    Released != "",
    Title != "",
    Transcript != ""
  )

# Clean the data
df <- df %>% mutate(
  Views = map_dbl(Views, cleanViews),
  Subscribers = map_dbl(Subscribers, cleanSubscribers),
  Length = map_dbl(Length, cleanLength),
  Released = map_dbl(Released, cleanReleased)
)

df %>% dplyr::select(URL, Channel, Views, Subscribers, Released, Length) %>% head(10)

# Save for future use
write_csv(df, "cleaned_data.csv")

```

After cleaning, Views, Subscribers, Released and Length are numbers, while Views, Subscribers are in K and Released and Length are in minute.

## Data Discovery / Diagnostics for Linear Regression

```
df <- read_csv("cleaned_data.csv")

## Rows: 2098 Columns: 11

## -- Column specification -----
## Delimiter: ","
## chr (6): Id, Channel, Title, URL, Category, Transcript
## dbl (5): Subscribers, CC, Released, Views, Length

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

df$CC <- as.factor(df$CC)
df$Category <- as.factor(df$Category)
df$Subscribers <- as.numeric(df$Subscribers)

head(df)

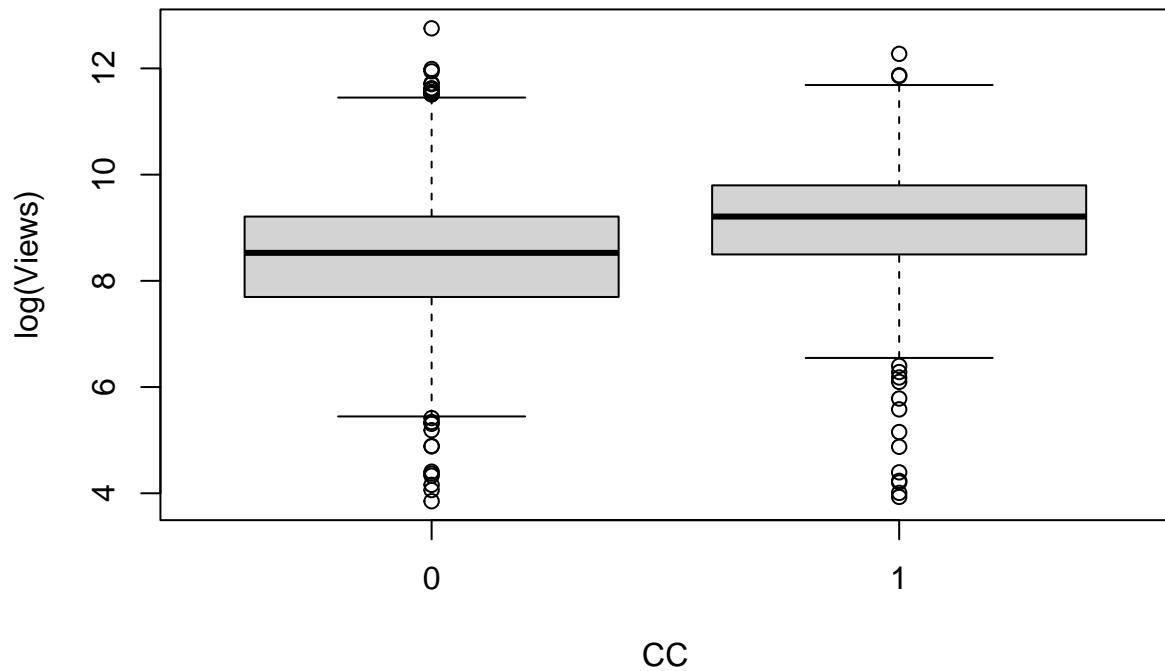
## # A tibble: 6 x 11
##   Id     Channel Subscribers Title CC      URL    Released Views Category Transcript
##   <chr> <chr>       <dbl> <chr> <fct> <chr>    <dbl> <dbl> <fct>   <chr>
## 1 FozC~ JRE Cl~       6280 Form~ 0     http~     24  7900 Blog      "the Joe ~
## 2 Iugc~ Munchi~       4590 The ~ 0     http~     24 11000 Food      "if you w~
## 3 JiEO~ Parks ~       282 Ron ~ 0     http~     36  2300 Enterta~ "April wh~
## 4 1T4X~ Vsauce        17400 What~ 1     http~    108 21000 Science   "Hey, Vsa~
## 5 OZWG~ Doctor~        1590 The ~ 1     http~     84  8500 Enterta~ "Oh, what~
## 6 YiEj~ A&E          7930 Live~ 1     http~     24 14000 News      "[music p~

## # ... with 1 more variable: Length <dbl>

table(df$CC)

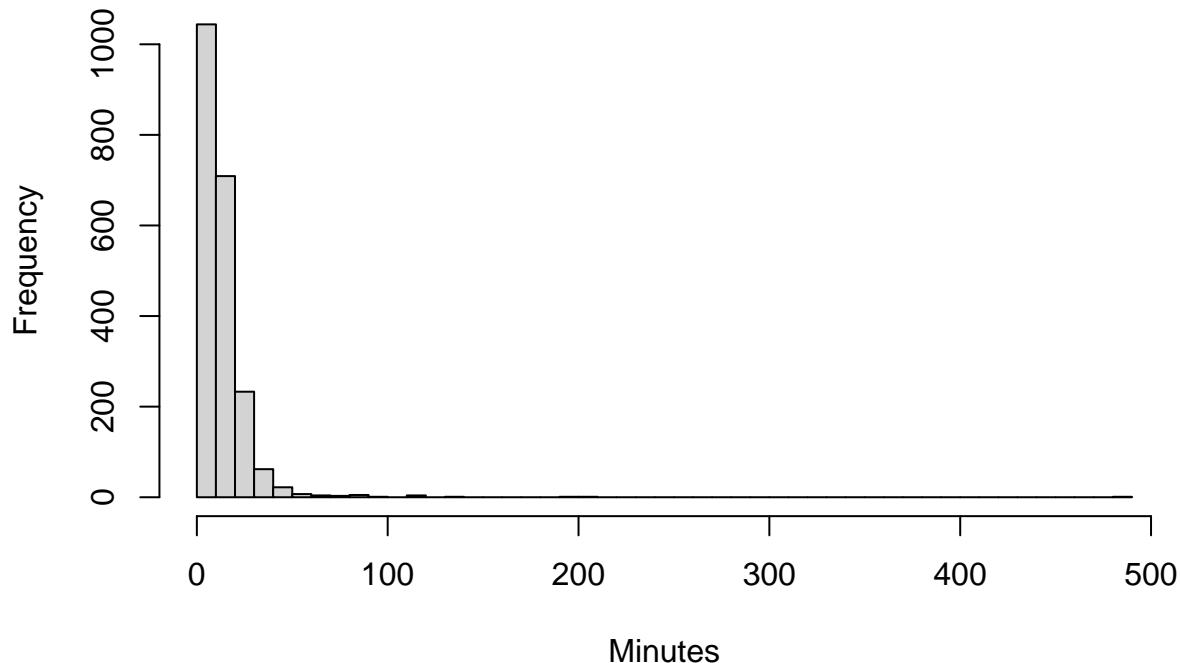
##
##     0     1
## 1094 1004

boxplot(log(Views) ~ CC, data = df)
```



```
hist(df$Length, xlab = "Minutes", breaks = 50)
```

## Histogram of df\$Length



```
summary(df$Length)
```

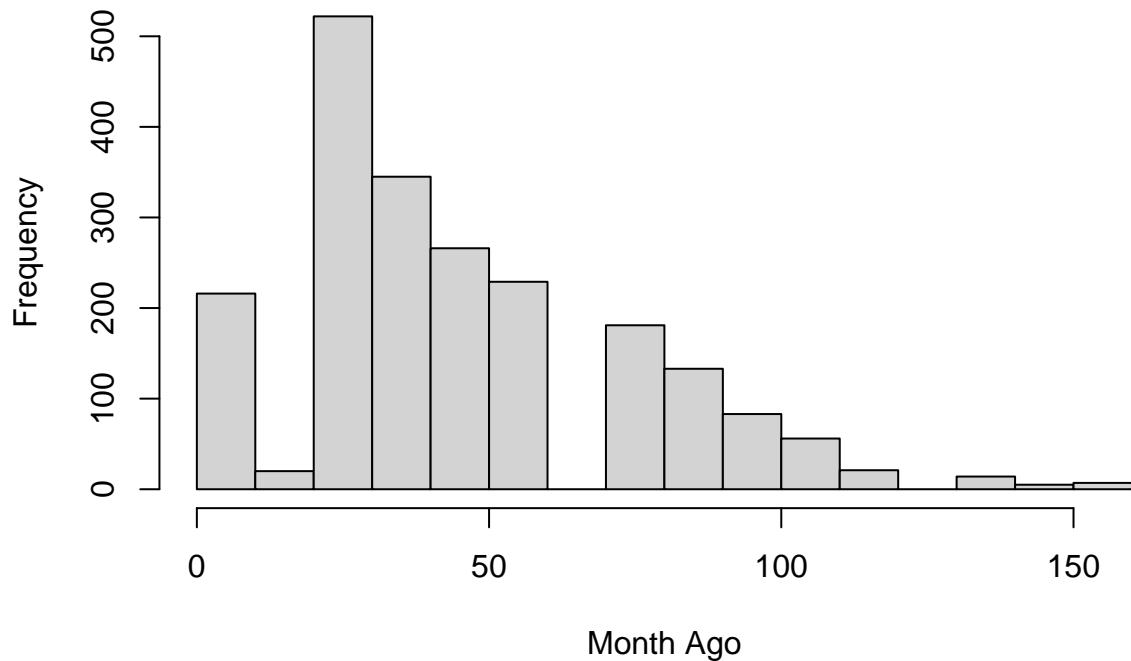
```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##     1.00    6.00   11.00   13.55   17.00  486.00
```

```
summary(df$Released)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##     2.00   24.00   36.00   46.46   60.00  156.00
```

```
hist(df$Released, xlab = "Month Ago")
```

## Histogram of df\$Released

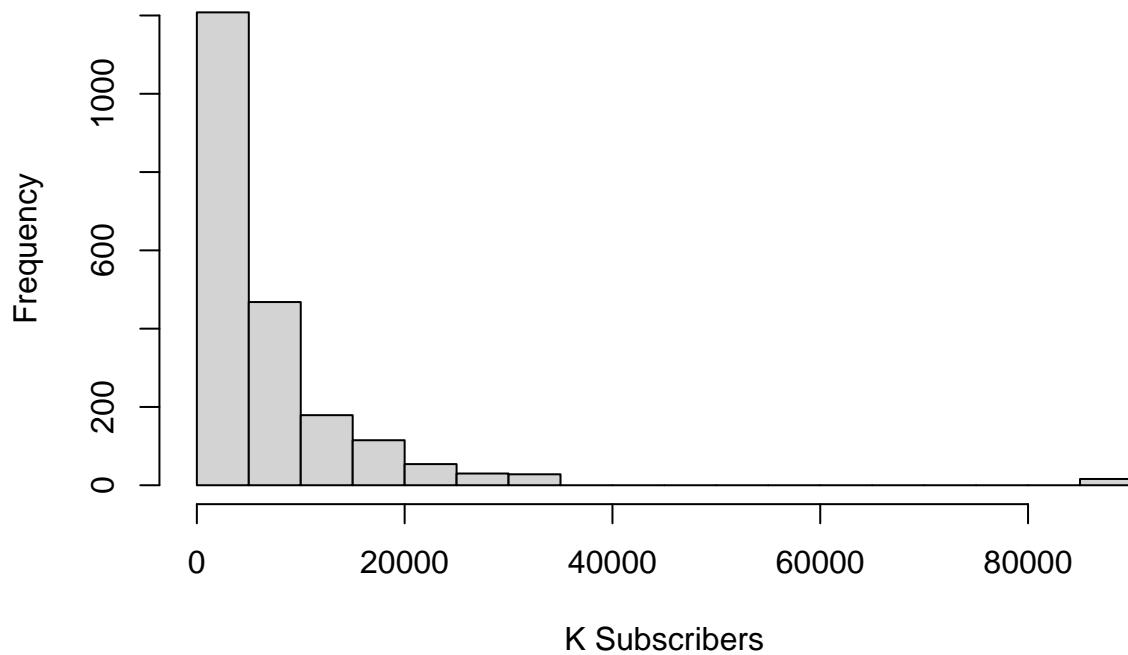


```
summary(df$Subscribers)
```

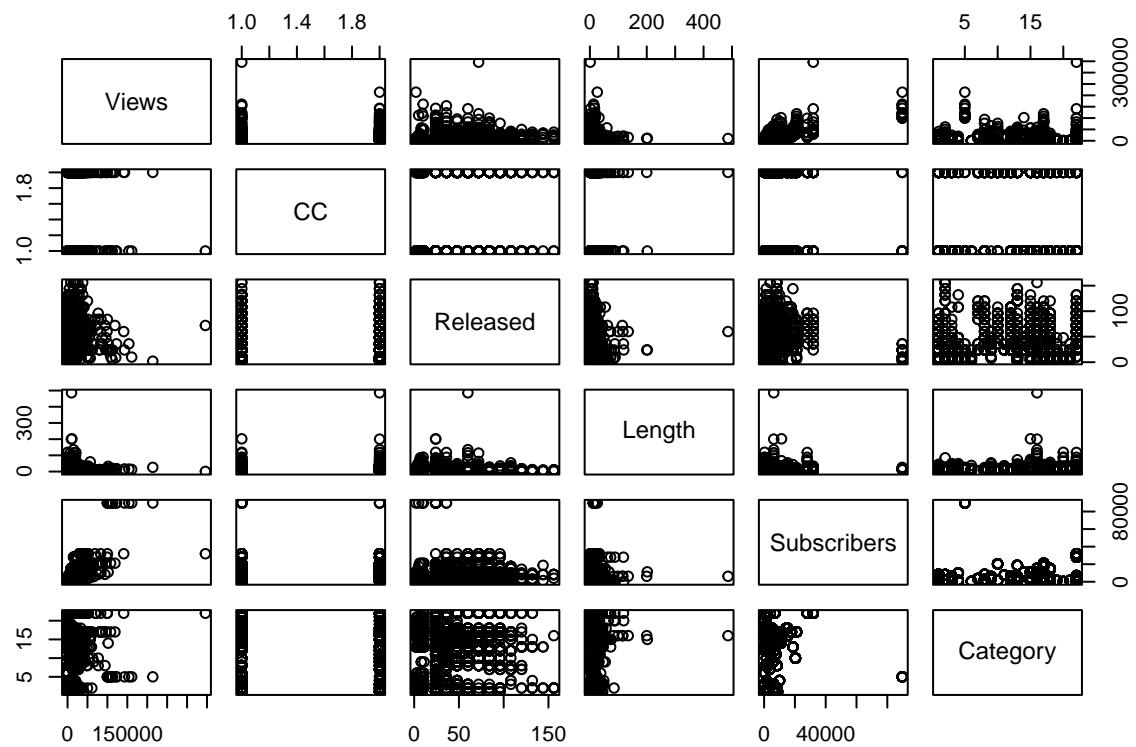
```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##      179     1730    4010     6871    8240   89700
```

```
hist(df$Subscribers, xlab = "K Subscribers")
```

## Histogram of df\$Subscribers

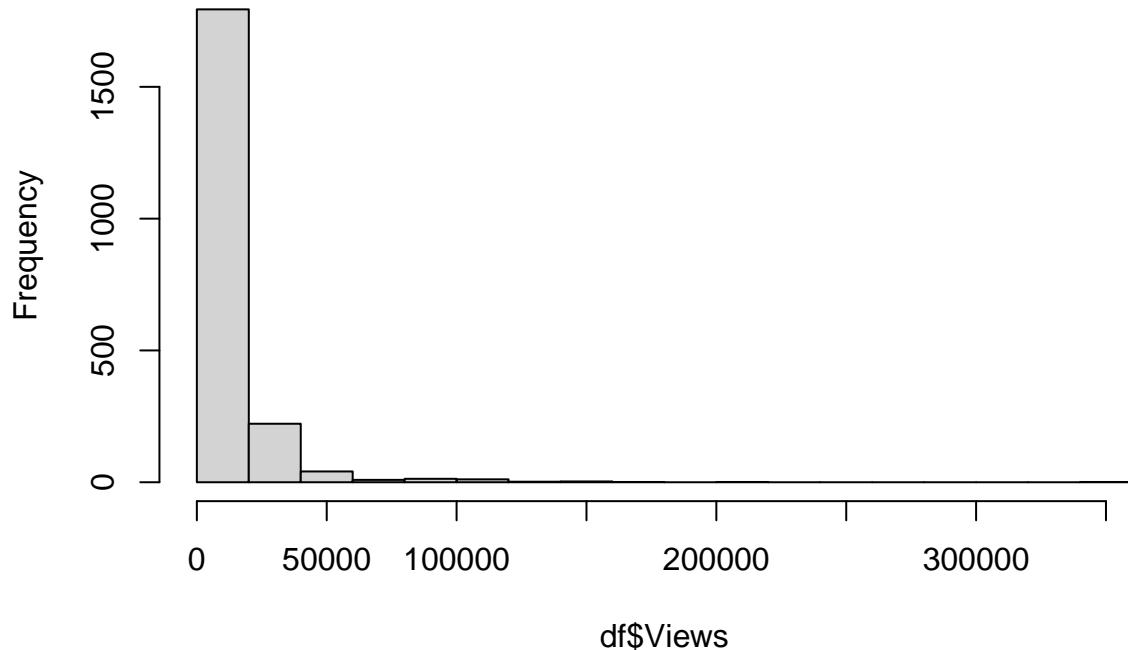


```
pairs(Views ~ CC + Released + Length + Subscribers + Category, data=df)
```



```
hist(df$Views)
```

### Histogram of df\$Views



From the diagnostics, Views, Subscribers, Released and Length need log transformation.

### Sentiment Analysis / Text mining

TODO by Xinyi

```
df_script <- df %>%
  dplyr::select(Id, Title, Transcript)
head(df_script)

## # A tibble: 6 x 3
##   Id      Title          Transcript
##   <chr>   <chr>          <chr>
## 1 FozCklixj-w Former CIA Agent Breaks Down~ "the Joe Rogan experience well how ~
## 2 IugcIAAZJ2M The Iconic $1 Pizza Slice of~ "if you want good pizza come to st ~
## 3 JiE06F8i0eU Ron Swanson: The Papa of Paw~ "April where have you been over two~
## 4 1T4XMNN4bNM What's The Most Dangerous Pl~ "Hey, Vsauce. Michael here. 93% of ~
## 5 OZWGeidvrJw The Doctor Defeats the Abzor~ "Oh, what's the matter?\nHave you g~
## 6 YiEj9mrqTNO Live PD: Most Viewed Moments~ "[music playing] We'll be on Lavern~

# just use this code to watch the video to check the transcript
df %>%
  filter(Title == "Former CIA Agent Breaks Down Jeffrey Epstein Case")
```

```

data("stop_words")
custom_stop_words <- rbind(stop_words, c("_", "custom"))

#bigram
bigrams_separated <- df_script %>%
  group_by(Id) %>%
  # unnest Transcript in bigram format
  unnest_tokens(bigram, Transcript, token = "ngrams", n = 2) %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>% # separate bigram
  filter(!word1 %in% custom_stop_words$word) %>% # filter out all the stop words
  filter(!word2 %in% custom_stop_words$word)

bigrams_united <- bigrams_separated %>%
  unite(bigram, word1, word2, sep = " ") # unite words back together

head(bigrams_separated)

# Using bigrams to provide context in sentiment analysis
# not in presentation!!
negation_words <- c("not", "no", "never", "without")

bigrams_separated %>%
  filter(word1 %in% negation_words) %>%
  inner_join(get_sentiments("afinn"), by = c(word2 = "word")) %>%
  count(word1, word2, value, sort = TRUE)

df_word <- df %>%
  group_by(Id) %>%
  unnest_tokens(word, Transcript) %>%
  anti_join(custom_stop_words) %>%
  count(word, sort = TRUE) %>%
  mutate(total = sum(n)) %>%
  ungroup()

## Joining, by = "word"

#inner_join(get_sentiments("afinn")) # %>%
#summarise(sentiment = sum(value))
#mutate(total = sum(word)) %>%
#mutate(perc = round(n/total, 2))

head(df_word)

## # A tibble: 6 x 4
##   Id      word      n total
##   <chr>    <chr>    <int> <int>
## 1 prd2RfhF1tM president  566 19722
## 2 prd2RfhF1tM trump     345 19722
## 3 FPs_1U01KoI president  275  9702
## 4 prd2RfhF1tM people    259 19722
## 5 fTgm36y884c cheese    197  2817
## 6 FPs_1U01KoI complaint  187  9702

```

```

df_title_word <- df %>%
  group_by(Id) %>%
  unnest_tokens(word, Title) %>%
  anti_join(custom_stop_words) %>%
  count(word, sort = TRUE) %>%
  mutate(total = sum(n)) %>%
  ungroup()

## Joining, by = "word"

#inner_join(get_sentiments("afinn")) #>%
#summarise(sentiment = sum(value))
#mutate(total = sum(word)) %>%
#mutate(perc = round(n/total, 2))

head(df_title_word)

## # A tibble: 6 x 4
##   Id      word     n total
##   <chr>   <chr> <int> <int>
## 1 DLjJwW1lFxI doctor     4     8
## 2 s5GfhGFVCFE 20          4     7
## 3 0e71KwxE5Fk doctor     3    10
## 4 GLSPub4ydiM core       3     7
## 5 uB_p1Pyv1ps paul       3    11
## 6 ZgyUOLyWZ9M con        3     9

```

## Below codes:

use tf-idf to find the importance of a word in the transcript, then times it to a word's afinn score, and sum up all the words' score to a video afinn\_score.

Will use “afinn\_score” to represent the sentiment score in the regression modeling!

## Transcript sentiment

```

df_afinn <- df_word %>%
  left_join(get_sentiments("afinn")) %>%
  #mutate(afinn_score = sum(value)) %>%
  #mutate(perc = round(n/total, 2)) %>%
  group_by(Id) %>%
  bind_tf_idf(word, Id, n) %>%
  mutate(value = ifelse(is.na(value), 0, value)) %>%
  mutate(afinn_score = sum(value*tf_idf)) %>%
  ungroup()

## Joining, by = "word"

```

```

#filter>Title == "2018 Jeep Trackhawk Review - The SUV That's Quicker Than a Supercar")

head(df_afinn)

## # A tibble: 6 x 9
##   Id      word     n total value    tf    idf  tf_idf afinn_score
##   <chr>   <chr> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 prd2RfhF1tM president  566 19722    0 0.0287 3.03  0.0871  0.146
## 2 prd2RfhF1tM trump     345 19722    0 0.0175 3.74  0.0654  0.146
## 3 FPs_1U01KoI president  275 9702     0 0.0283 3.03  0.0860  0.0362
## 4 prd2RfhF1tM people   259 19722    0 0.0131 0.386  0.00507 0.146
## 5 fTgm36y884c cheese    197 2817     0 0.0699 2.49  0.174   0.167
## 6 FPs_1U01KoI complaint 187 9702     0 0.0193 4.39  0.0846  0.0362

df_afinn <- df_afinn %>%
  dplyr::select(Id, afinn_score) %>%
  unique() %>%
  ungroup()

df_afinn

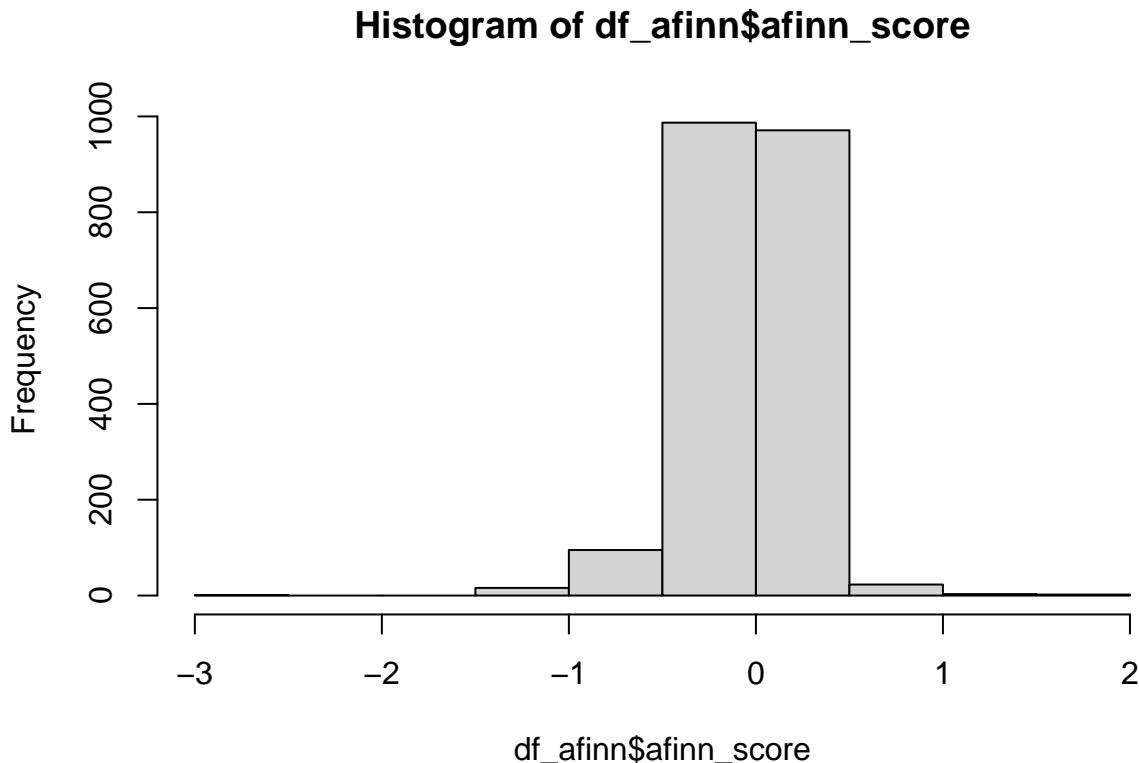
## # A tibble: 2,098 x 2
##   Id      afinn_score
##   <chr>       <dbl>
## 1 prd2RfhF1tM 0.146
## 2 FPs_1U01KoI 0.0362
## 3 fTgm36y884c 0.167
## 4 qWAagS_MANg 0.0225
## 5 YeFzkC2awTM -0.00155
## 6 p3qvj9h0_Bo 0.0548
## 7 fbjYkPKRm-8 0.0285
## 8 ZRuSS0iiFyo -0.285
## 9 hc3TEaT3WHA 0.0613
## 10 4VTOp1Ll2BM 0.155
## # ... with 2,088 more rows

summary(df_afinn$afinn_score)

##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
## -2.68050 -0.17631 -0.01230 -0.04816  0.10676  1.72980

```

```
hist(df_afinn$afinn_score)
```



## Title sentiment

```
df_title_afinn <- df_title_word %>%
  left_join(get_sentiments("afinn")) %>%
  #mutate(afinn_score = sum(value)) %>%
  #mutate(perc = round(n/total, 2)) %>%
  group_by(Id) %>%
  bind_tf_idf(word, Id, n) %>%
  mutate(value = ifelse(is.na(value), 0, value)) %>%
  mutate(afinn_title_score = sum(value*tf_idf)) %>%
  ungroup()

## Joining, by = "word"

#filter(Title == "2018 Jeep Trackhawk Review - The SUV That's Quicker Than a Supercar")

head(df_title_afinn)

## # A tibble: 6 x 9
```

```

##   Id      word      n total value      tf     idf tf_idf afinn_title_score
##   <chr>    <chr> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 DLjJwW1lFxI doctor     4     8     0 0.5    4.39    2.19          0
## 2 s5GfhGFVCFE 20        4     7     0 0.571   5.70    3.26          0
## 3 0e71KwxE5Fk doctor     3    10     0 0.3    4.39    1.32          0
## 4 GLSPub4ydiM core      3     7     0 0.429   6.55    2.81          0
## 5 uB_p1Pyv1ps paul      3    11     0 0.273   5.01    1.37   -1.39
## 6 ZgyUOLyWZ9M con       3     9     0 0.333   6.95    2.32          0

```

```

df_title_afinn <- df_title_afinn %>% dplyr::select(Id, afinn_title_score) %>%
  unique() %>%
  ungroup()

```

```
head(df_title_afinn)
```

```

## # A tibble: 6 x 2
##   Id      afinn_title_score
##   <chr>    <dbl>
## 1 DLjJwW1lFxI          0
## 2 s5GfhGFVCFE          0
## 3 0e71KwxE5Fk          0
## 4 GLSPub4ydiM          0
## 5 uB_p1Pyv1ps         -1.39
## 6 ZgyUOLyWZ9M          0

```

```
summary(df_title_afinn$afinn_title_score)
```

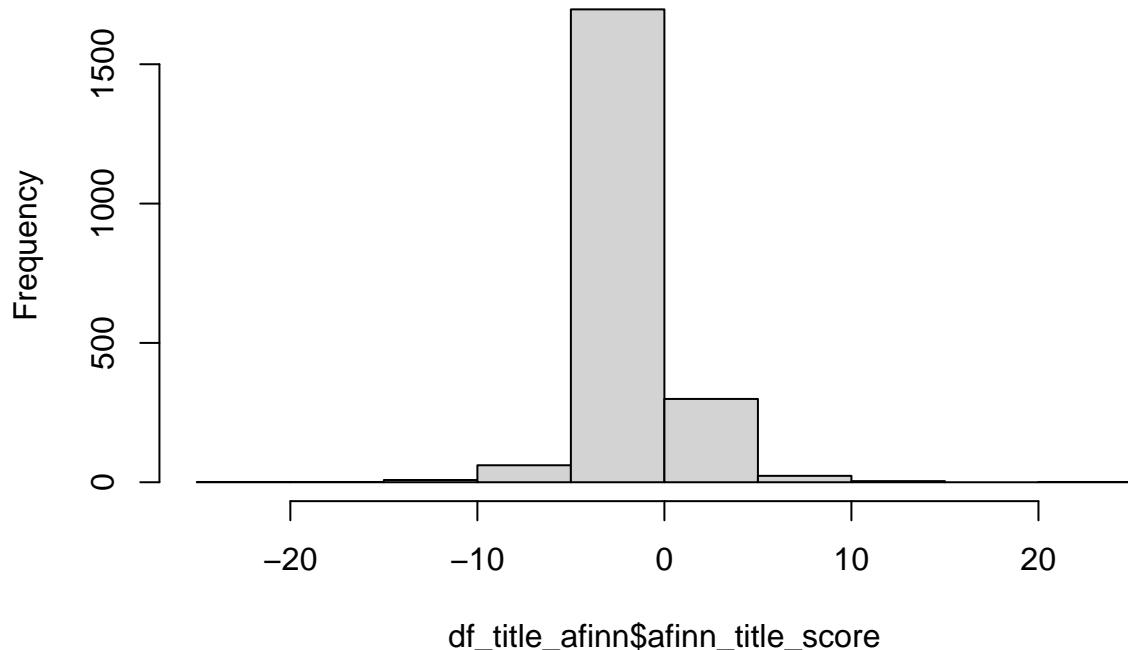
```

##      Min.    1st Qu.   Median    Mean    3rd Qu.   Max.
## -20.8625    0.0000   0.0000  -0.2664   0.0000  22.9419

```

```
hist(df_title_afinn$afinn_title_score)
```

## Histogram of df\_title\_afinn\$afinn\_title\_score



```
df1 <- df %>%
  left_join(df_afinn) %>%
  left_join(df_title_afinn) %>%
  mutate(
    afinn_title_score = ifelse(is.na(afinn_title_score), 0, afinn_title_score)
  ) %>%
  unique()
```

```
## Joining, by = "Id"
## Joining, by = "Id"
```

```
head(df1)
```

```
## # A tibble: 6 x 13
##   Id     Channel Subscribers Title CC     URL   Released Views Category Transcript
##   <chr> <chr>       <dbl> <chr> <fct> <chr>   <dbl> <dbl> <fct>   <chr>
## 1 FozC~ JRE Cl~        6280 Form~ 0      http~     24  7900 Blog     "the Joe ~
## 2 Iugc~ Munchi~        4590 The ~ 0      http~     24 11000 Food     "if you w~
## 3 JiE0~ Parks ~        282 Ron ~ 0      http~     36  2300 Enterta~ "April wh~
## 4 1T4X~ Vsauce         17400 What~ 1      http~    108 21000 Science  "Hey, Vsa~
## 5 OZWG~ Doctor~        1590 The ~ 1      http~     84  8500 Enterta~ "Oh, what~
## 6 YiEj~ A&E           7930 Live~ 1      http~     24 14000 News     "[music p~
## # ... with 3 more variables: Length <dbl>, afinn_score <dbl>,
## #   afinn_title_score <dbl>
```

```
# Save for future use
write_csv(df1, "cleaned_data_with_sentiment.csv")
```

## Preparation for Cross-Validation

Randomly split the data set in a 70% training and 30% test set. Make sure to use `set.seed()` so that your results are reproducible

```
df1 <- read_csv("cleaned_data_with_sentiment.csv")

## Rows: 2098 Columns: 13

## -- Column specification -----
## Delimiter: ","
## chr (6): Id, Channel, Title, URL, Category, Transcript
## dbl (7): Subscribers, CC, Released, Views, Length, afinn_score, afinn_title_...
## 
## i Use 'spec()' to retrieve the full column specification for this data.
## i Specify the column types or set 'show_col_types = FALSE' to quiet this message.

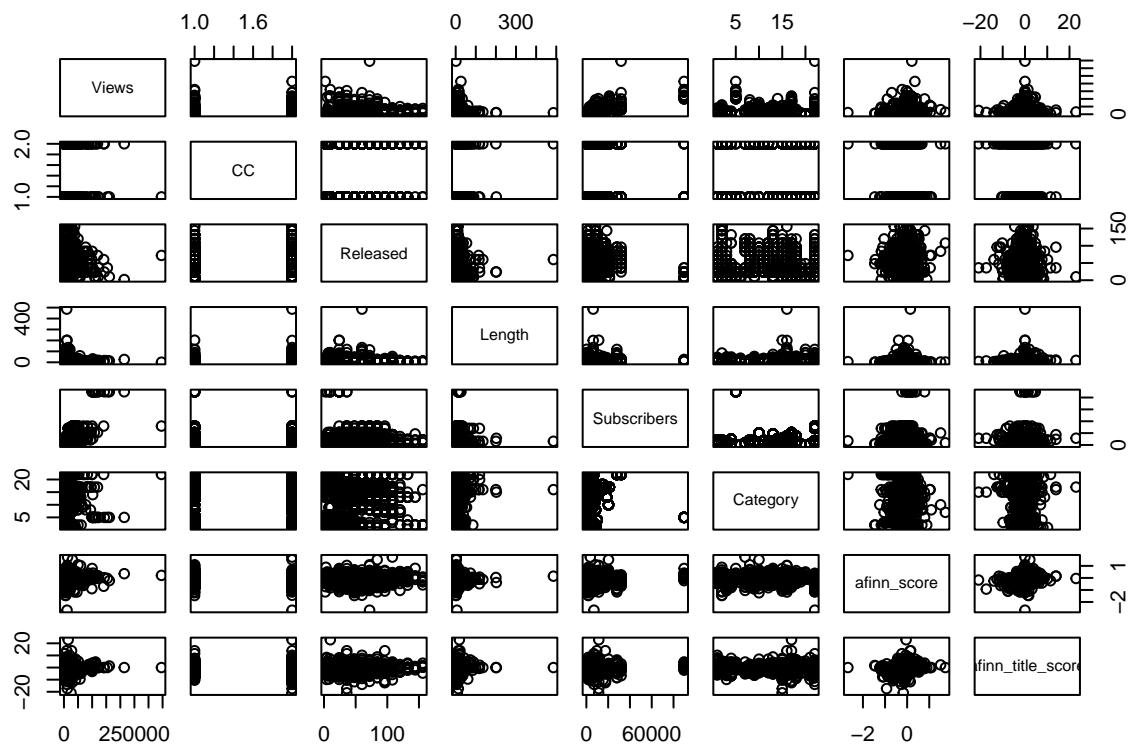
df1$CC <- as.factor(df1$CC)
df1$Category <- as.factor(df1$Category)
df1$Subscribers <- as.numeric(df1$Subscribers)

set.seed(652)
n <- nrow(df1)
train_index <- sample(1:n, round(0.7*n))
df_train <- df1[train_index,]
df_test <- df1[-train_index,]

# function to compute RMSE
RMSE <- function(y, y_hat) {
  sqrt(mean((y - y_hat)^2))
}
```

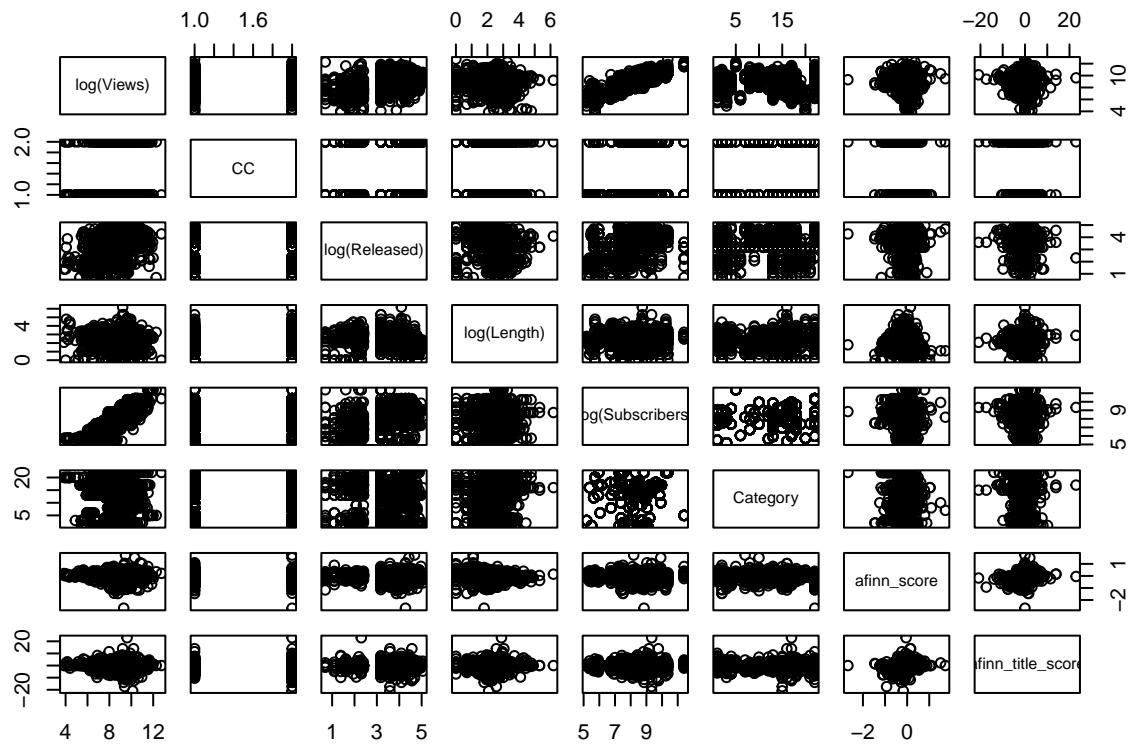
## Linear Regression

```
# original response and predictors
pairs(Views ~ CC + Released +
      Length + Subscribers + Category +
      afinn_score + afinn_title_score,
      data=df1)
```



From the plot above, we didn't see any obvious strong correlation between the predictors.

```
# log response and other predictors that are right skewed
pairs(log(Views) ~ CC + log(Released) +
      log(Length) + log(Subscribers) + Category +
      afinn_score + afinn_title_score,
      data=df1)
```



From the plot above, we didn't see any obvious strong correlation between the predictors.

```
round(cor(df1[, c(3,7,11,12,13)]), 2)
```

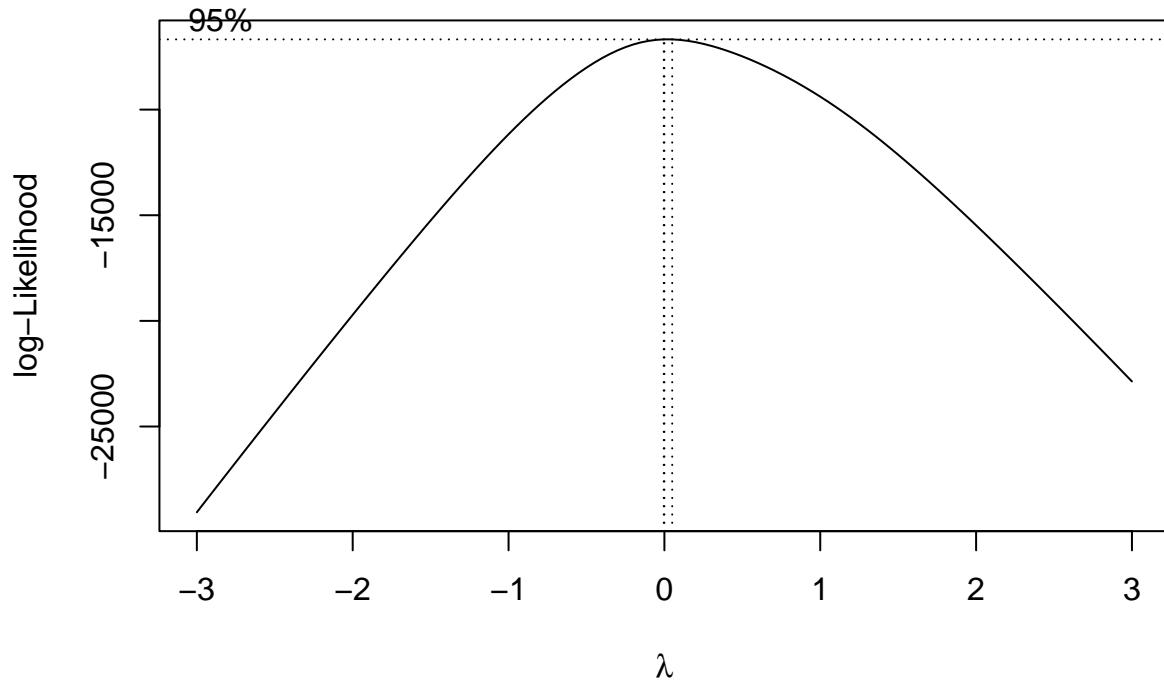
	Subscribers	Released	Length	afinn_score	afinn_title_score
## Subscribers	1.00	0.09	0.02	-0.02	0.01
## Released	0.09	1.00	-0.18	-0.05	0.03
## Length	0.02	-0.18	1.00	0.03	0.03
## afinn_score	-0.02	-0.05	0.03	1.00	0.22
## afinn_title_score	0.01	0.03	0.03	0.22	1.00

The correlation table also indicates this.

```
plot(log(Views) ~ afinn_score, data = df1)
```

```
plot(log(Views) ~ afinn_title_score, data = df1)
```

```
boxcox(Views ~ CC + log(Released) + log(Length) + log(Subscribers) + Category + afinn_score + afinn_title_score, data=df1, lambda = seq(-3, 3, by = 0.05))
```



```

summary(powerTransform(Views ~
  CC + log(Released) + log(Length) +
  log(Subscribers) + Category + afinn_score + afinn_title_score,
  data=df1))

## bcPower Transformation to Normality
##   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
##   Y1      0.021        0.02       7e-04     0.0412
##
## Likelihood ratio test that transformation parameter is equal to 0
## (log transformation)
##           LRT df      pval
## LR test, lambda = (0) 4.15773  1 0.041445
##
## Likelihood ratio test that no transformation is needed
##           LRT df      pval
## LR test, lambda = (1) 5428.034  1 < 2.22e-16

lm_full <- lm(Views ~ CC + Released + Length + Subscribers + Category + afinn_score+ afinn_title_score,
summary(lm_full)

##
## Call:
## lm(formula = Views ~ CC + Released + Length + Subscribers + Category +
##     afinn_score + afinn_title_score, data = df1)

```

```

##
## Residuals:
##      Min     1Q Median     3Q    Max
## -25663  -4088   -883  1907 299360
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)               5.013e+02  1.034e+03  0.485  0.627801
## CC1                      5.904e+02  5.813e+02  1.016  0.309909
## Released                  1.656e+00  9.779e+00  0.169  0.865512
## Length                   -2.443e+01  1.587e+01 -1.539  0.123935
## Subscribers                1.489e+00  4.445e-02 33.487 < 2e-16 ***
## CategoryAutomobile,Comedy  5.772e+03  1.647e+03  3.504  0.000468 ***
## CategoryBlog                 -8.700e+02  1.362e+03 -0.639  0.522927
## CategoryBlog,Comedy        -2.030e+03  1.975e+03 -1.028  0.304224
## CategoryBlog,Entertainment -1.060e+04  4.895e+03 -2.165  0.030486 *
## CategoryBlog,Science        -3.780e+02  3.591e+03 -0.105  0.916190
## CategoryComedy              1.086e+04  2.341e+03  4.637  3.75e-06 ***
## CategoryComedy,Entertainment 9.045e+03  1.555e+03  5.815  7.02e-09 ***
## CategoryComedy,Informative  1.114e+04  1.946e+03  5.725  1.19e-08 ***
## CategoryEntertainment       3.926e+03  1.848e+03  2.124  0.033754 *
## CategoryEntertainment,Blog  2.765e+03  2.393e+03  1.156  0.247943
## CategoryEntertainment,Comedy 6.283e+03  1.564e+03  4.017  6.12e-05 ***
## CategoryFood                 2.560e+03  1.180e+03  2.170  0.030109 *
## CategoryFood,Entertainment  8.604e+03  2.548e+03  3.377  0.000745 ***
## CategoryInformative          2.401e+02  1.251e+03  0.192  0.847864
## CategoryNews                  1.351e+03  1.348e+03  1.002  0.316437
## CategoryScience                1.968e+03  1.181e+03  1.667  0.095759 .
## CategoryTech                  -3.942e+03  1.274e+03 -3.094  0.002001 **
## CategoryTech,Comedy           -6.025e+02  2.617e+03 -0.230  0.817916
## CategoryTech,Informative      -4.183e+02  2.813e+03 -0.149  0.881818
## CategoryTech,News              -1.998e+03  2.942e+03 -0.679  0.497212
## CategoryVideoGames            -1.694e+03  1.239e+03 -1.367  0.171766
## afinn_score                  1.187e+03  9.982e+02  1.189  0.234713
## afinn_title_score             -1.251e+02  1.167e+02 -1.073  0.283554
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11470 on 2070 degrees of freedom
## Multiple R-squared:  0.5998, Adjusted R-squared:  0.5946
## F-statistic: 114.9 on 27 and 2070 DF,  p-value: < 2.2e-16

lm1 <- lm(log(Views) ~ CC + log(Released) + log(Length) + log(Subscribers) + Category + afinn_score+ af
summary(lm1)

```

```

##
## Call:
## lm(formula = log(Views) ~ CC + log(Released) + log(Length) +
##     log(Subscribers) + Category + afinn_score + afinn_title_score,
##     data = df1)
##
## Residuals:
##      Min     1Q Median     3Q    Max
## -1.28291 -0.34836 -0.06359  0.29517  2.62433

```

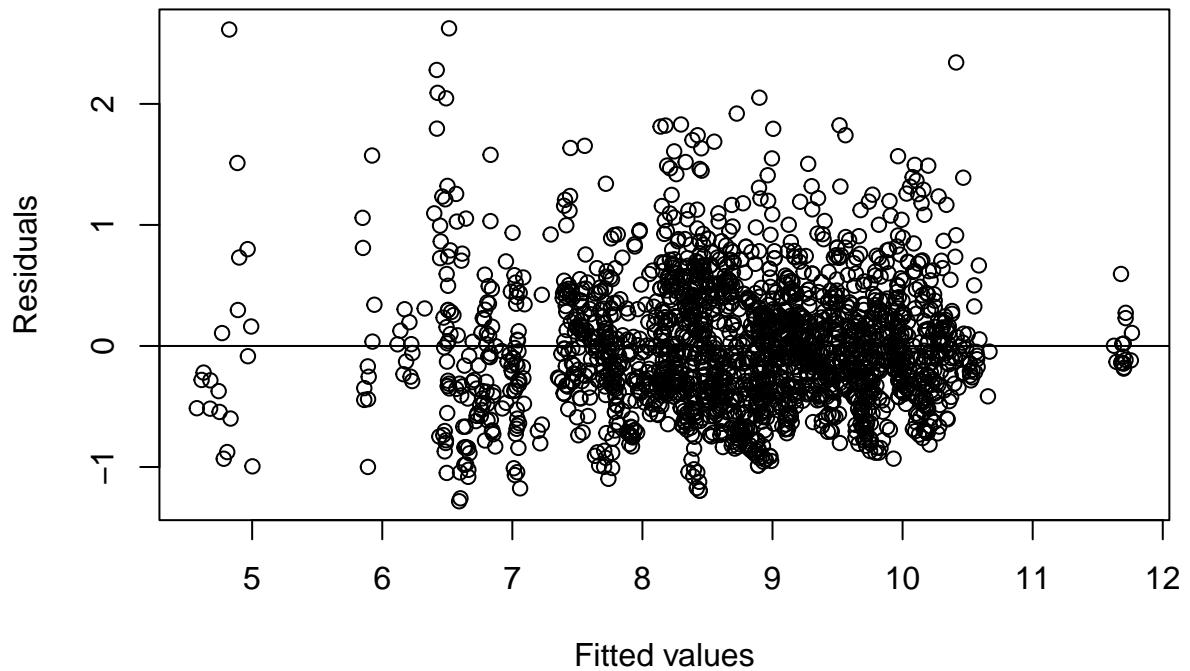
```

## 
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)           1.994258   0.115062 17.332 < 2e-16 ***
## CC1                  0.061293   0.027270  2.248 0.024702 *  
## log(Released)        0.010325   0.016986  0.608 0.543333    
## log(Length)          -0.081625   0.016780 -4.864 1.23e-06 *** 
## log(Subscribers)     0.828092   0.012171 68.038 < 2e-16 *** 
## CategoryAutomobile,Comedy 0.404303   0.076396  5.292 1.34e-07 *** 
## CategoryBlog          -0.218388   0.063018 -3.465 0.000540 *** 
## CategoryBlog,Comedy   -0.133908   0.091557 -1.463 0.143737    
## CategoryBlog,Entertainment 0.458470   0.147371  3.111 0.001890 **  
## CategoryBlog,Science   -0.626816   0.167315 -3.746 0.000184 *** 
## CategoryComedy         0.950982   0.108783  8.742 < 2e-16 *** 
## CategoryComedy,Entertainment 0.832074   0.072040 11.550 < 2e-16 *** 
## CategoryComedy,Informative 0.785949   0.089692  8.763 < 2e-16 *** 
## CategoryEntertainment  0.393832   0.085010  4.633 3.83e-06 *** 
## CategoryEntertainment,Blog 0.321985   0.110675  2.909 0.003661 **  
## CategoryEntertainment,Comedy 0.904802   0.072662 12.452 < 2e-16 *** 
## CategoryFood            0.295938   0.054581  5.422 6.58e-08 *** 
## CategoryFood,Entertainment 0.511487   0.118622  4.312 1.69e-05 *** 
## CategoryInformative      0.040345   0.057819  0.698 0.485392    
## CategoryNews             0.199804   0.062303  3.207 0.001362 **  
## CategoryScience          -0.049413   0.054384 -0.909 0.363669    
## CategoryTech             -0.583077   0.058816 -9.914 < 2e-16 *** 
## CategoryTech,Comedy      -0.381650   0.120859 -3.158 0.001612 **  
## CategoryTech,Informative -1.785302   0.132530 -13.471 < 2e-16 *** 
## CategoryTech,News         -1.015799   0.136331 -7.451 1.35e-13 *** 
## CategoryVideoGames        -0.202371   0.056352 -3.591 0.000337 *** 
## afinn_score              -0.057197   0.046152 -1.239 0.215366    
## afinn_title_score         -0.006662   0.005383 -1.238 0.215996    
## ---                        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.5293 on 2070 degrees of freedom
## Multiple R-squared:  0.8154, Adjusted R-squared:  0.813 
## F-statistic: 338.7 on 27 and 2070 DF,  p-value: < 2.2e-16
```

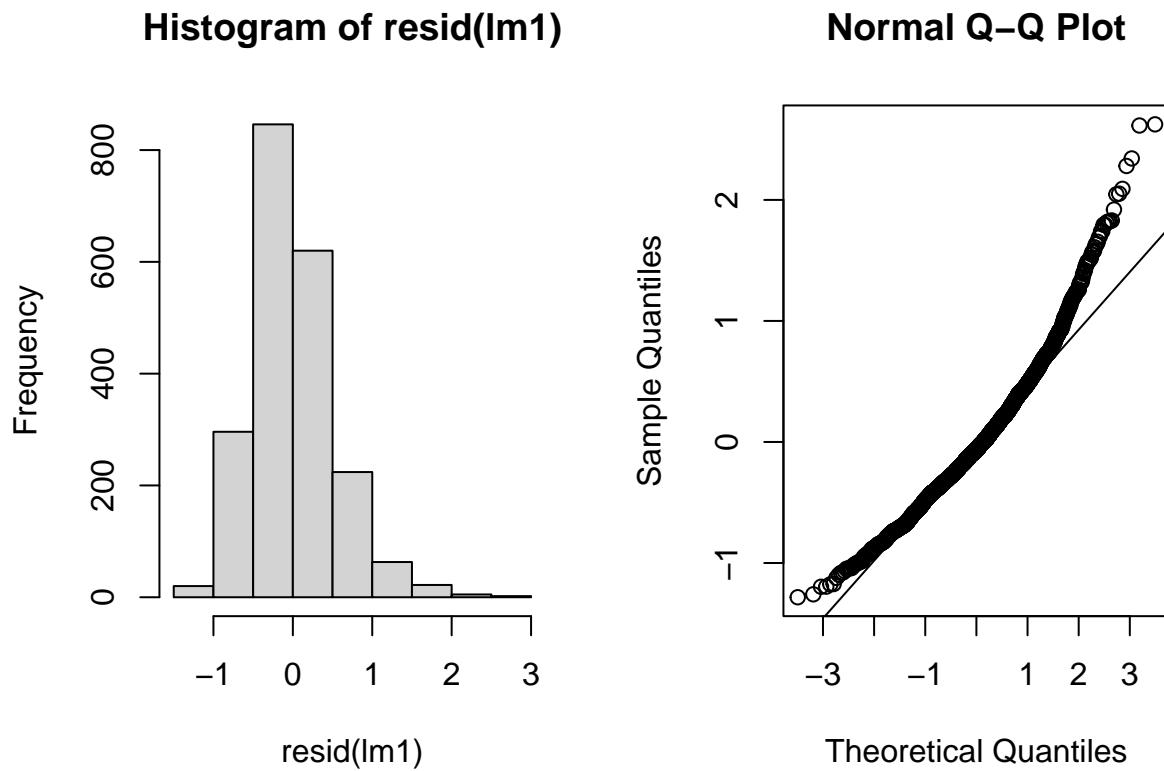
```

plot(predict(lm1), resid(lm1), xlab = "Fitted values", ylab = "Residuals")
abline(h=0)

```



```
par(mfrow=c(1,2))
hist(resid(lm1))
qqnorm(resid(lm1))
qqline(resid(lm1))
```



```
# remove log(Length) / Length
lm2 <- lm(log(Views) ~ CC + log(Length) + log(Subscribers) + Category + afinn_score + afinn_title_score
summary(lm2)
```

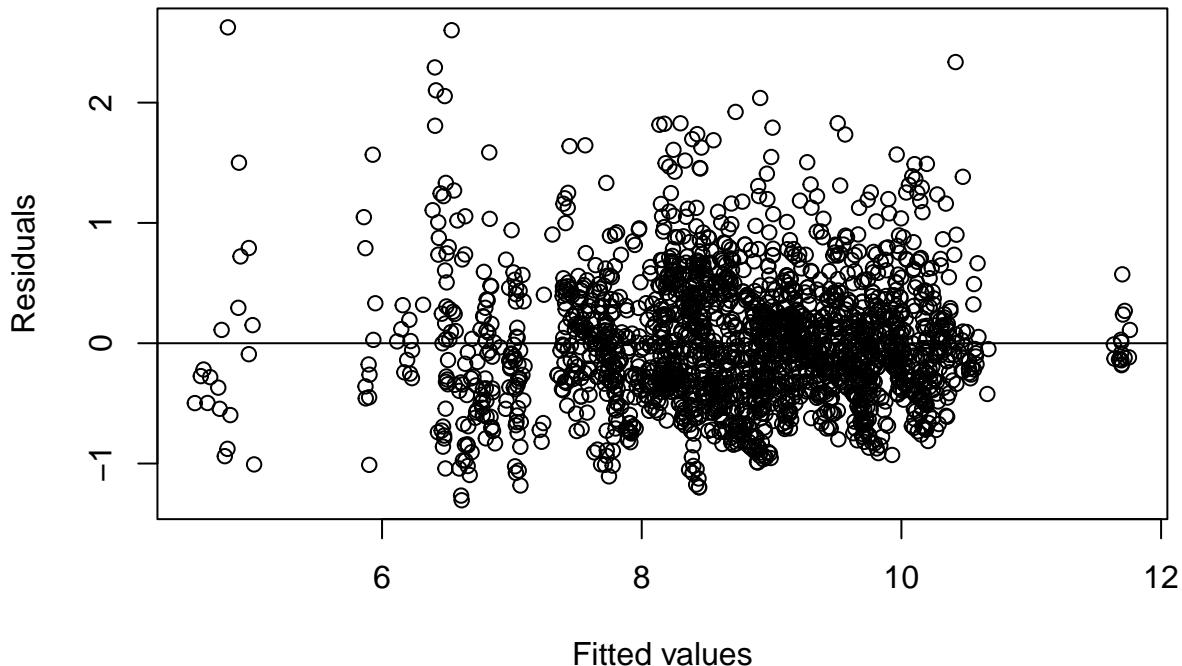
```
##
## Call:
## lm(formula = log(Views) ~ CC + log(Length) + log(Subscribers) +
##     Category + afinn_score + afinn_title_score, data = df1)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -1.30509 -0.34612 -0.06073  0.29173  2.62569
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2.019276  0.107433 18.796 < 2e-16 ***
## CC1                      0.062746  0.027160  2.310 0.020974 *
## log(Length)                -0.083559  0.016473 -5.072 4.28e-07 ***
## log(Subscribers)           0.830128  0.011700 70.953 < 2e-16 ***
## CategoryAutomobile,Comedy  0.402173  0.076304  5.271 1.50e-07 ***
## CategoryBlog                 -0.221634  0.062782 -3.530 0.000424 ***
## CategoryBlog,Comedy          -0.138616  0.091215 -1.520 0.128748
## CategoryBlog,Entertainment   0.444115  0.145445  3.053 0.002291 **
## CategoryBlog,Science          -0.638321  0.166216 -3.840 0.000127 ***
## CategoryComedy                0.953490  0.108689  8.773 < 2e-16 ***
```

```

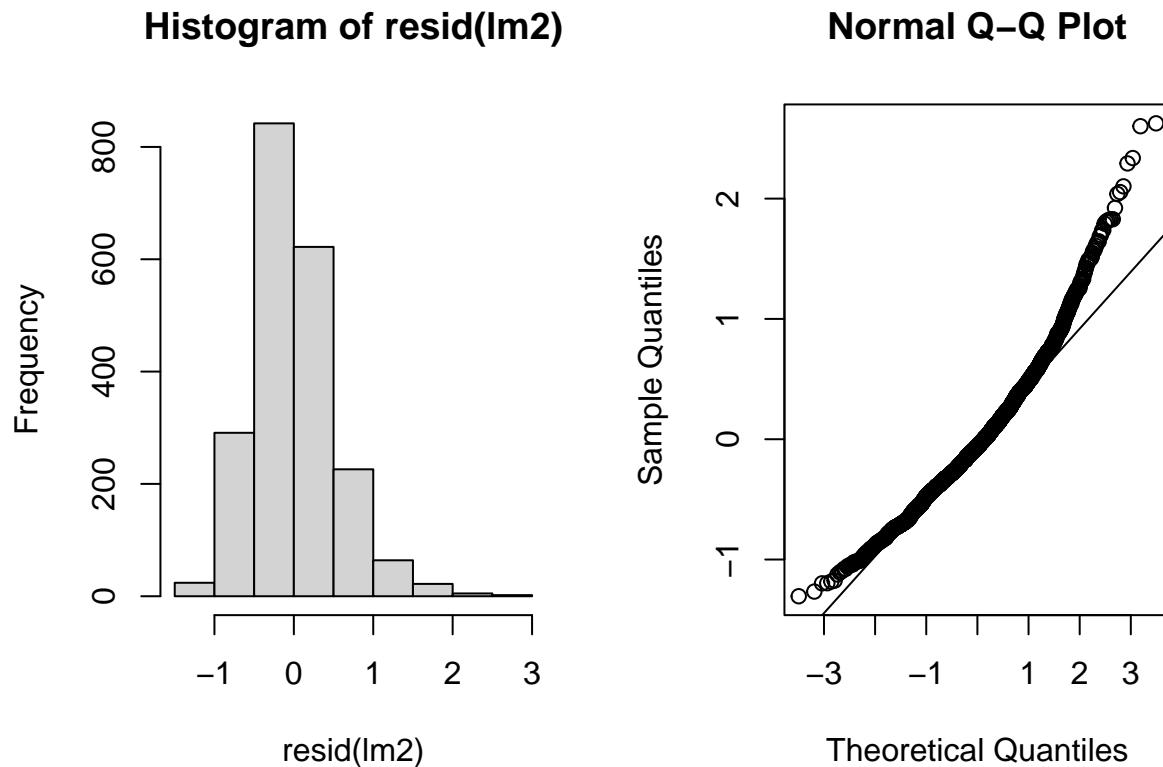
## CategoryComedy,Entertainment 0.834924 0.071877 11.616 < 2e-16 ***
## CategoryComedy,Informative 0.784121 0.089628 8.749 < 2e-16 ***
## CategoryEntertainment 0.396675 0.084868 4.674 3.14e-06 ***
## CategoryEntertainment,Blog 0.318365 0.110498 2.881 0.004003 **
## CategoryEntertainment,Comedy 0.908790 0.072354 12.560 < 2e-16 ***
## CategoryFood 0.293166 0.054382 5.391 7.81e-08 ***
## CategoryFood,Entertainment 0.510386 0.118590 4.304 1.76e-05 ***
## CategoryInformative 0.040077 0.057809 0.693 0.488215
## CategoryNews 0.197180 0.062144 3.173 0.001531 **
## CategoryScience -0.050974 0.054315 -0.938 0.348107
## CategoryTech -0.584057 0.058785 -9.935 < 2e-16 ***
## CategoryTech,Comedy -0.387360 0.120475 -3.215 0.001323 **
## CategoryTech,Informative -1.796723 0.131172 -13.697 < 2e-16 ***
## CategoryTech,News -1.025647 0.135345 -7.578 5.26e-14 ***
## CategoryVideoGames -0.198996 0.056069 -3.549 0.000395 ***
## afinn_score -0.056556 0.046132 -1.226 0.220360
## afinn_title_score -0.006585 0.005381 -1.224 0.221167
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ',' 1
##
## Residual standard error: 0.5293 on 2071 degrees of freedom
## Multiple R-squared: 0.8154, Adjusted R-squared: 0.8131
## F-statistic: 351.8 on 26 and 2071 DF, p-value: < 2.2e-16

plot(predict(lm2), resid(lm2), xlab = "Fitted values", ylab = "Residuals")
abline(h=0)

```



```
par(mfrow=c(1, 2))
hist(resid(lm2))
qqnorm(resid(lm2))
qqline(resid(lm2))
```



```
# remove log(Length) / Length and afinn_title_score
lm3 <- lm(log(Views) ~ CC + log(Length) + log(Subscribers) + afinn_score, data=df1)
summary(lm3)
```

```
##
## Call:
## lm(formula = log(Views) ~ CC + log(Length) + log(Subscribers) +
##     afinn_score, data = df1)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -3.09140 -0.47506 -0.04685  0.42968  2.82115
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.09716   0.11466 18.290 < 2e-16 ***
## CC1         0.11040   0.03093  3.570 0.000365 ***
## log(Length) -0.19559   0.01923 -10.170 < 2e-16 ***
## log(Subscribers) 0.85720   0.01332  64.347 < 2e-16 ***
## afinn_score -0.08173   0.05348 -1.528 0.126613
```

```

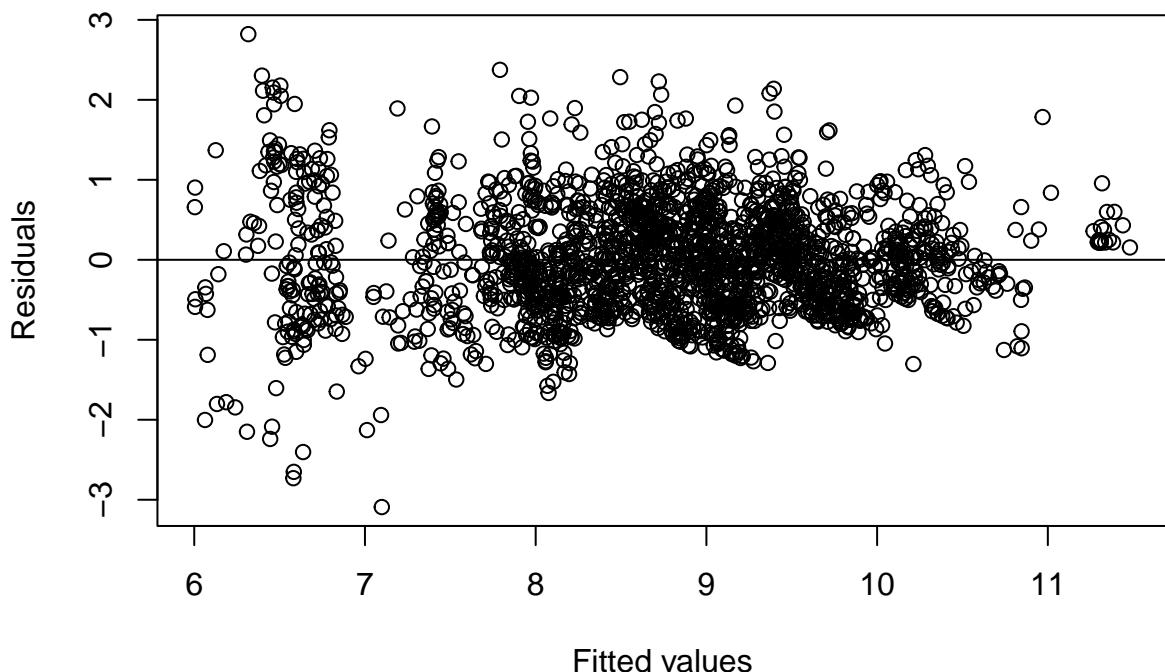
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6761 on 2093 degrees of freedom
## Multiple R-squared:  0.6956, Adjusted R-squared:  0.695
## F-statistic:  1196 on 4 and 2093 DF,  p-value: < 2.2e-16

```

```

plot(predict(lm3), resid(lm3), xlab = "Fitted values", ylab = "Residuals")
abline(h=0)

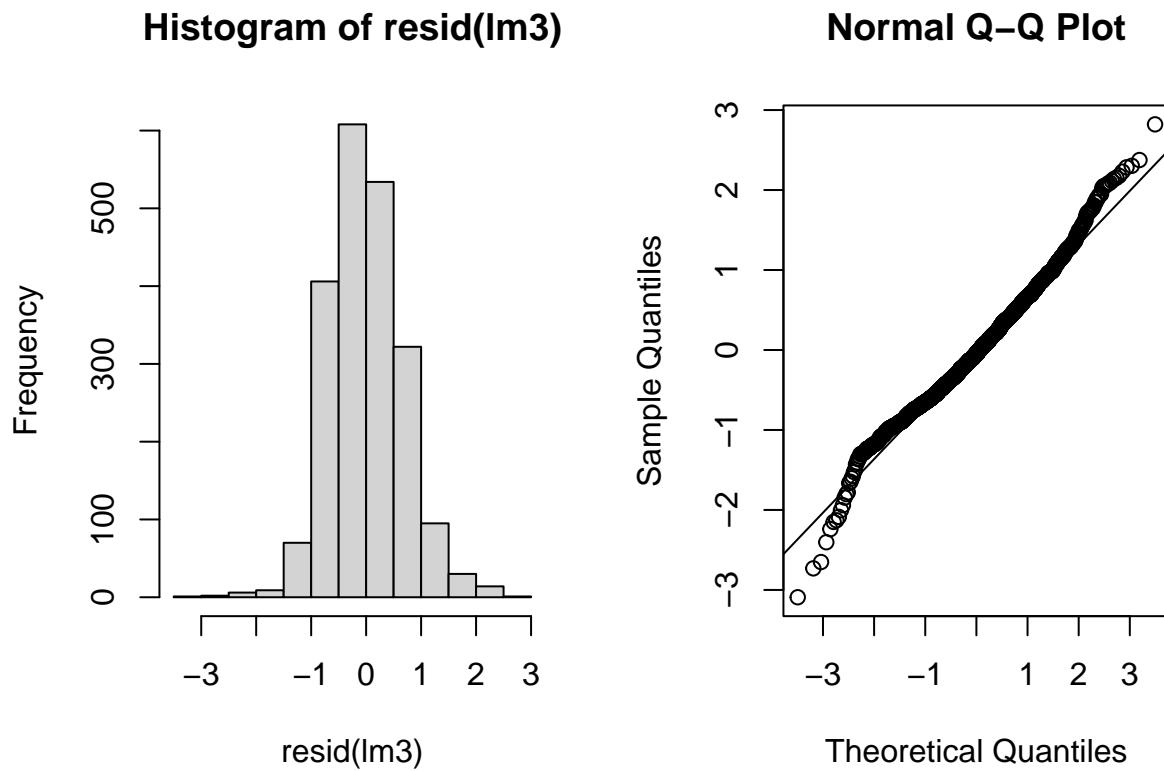
```



```

par(mfrow=c(1,2))
hist(resid(lm3))
qqnorm(resid(lm3))
qqline(resid(lm3))

```



```
# lm1 <- lm(log(Views) ~ CC + log(Released) + log(Length) + log(Subscribers) + Category + afinn_score +
lm4 <- step(lm1)
```

```
## Start: AIC=-2641.29
## log(Views) ~ CC + log(Released) + log(Length) + log(Subscribers) +
##   Category + afinn_score + afinn_title_score
##
##                               Df Sum of Sq    RSS      AIC
## - log(Released)           1    0.10  580.14 -2642.91
## - afinn_title_score       1    0.43  580.47 -2641.74
## - afinn_score              1    0.43  580.47 -2641.73
## <none>                      580.04 -2641.29
## - CC                       1    1.42  581.45 -2638.17
## - log(Length)              1    6.63  586.67 -2619.44
## - Category                 21   368.50 948.54 -1651.43
## - log(Subscribers)         1   1297.14 1877.18 -179.33
##
## Step: AIC=-2642.91
## log(Views) ~ CC + log(Length) + log(Subscribers) + Category +
##   afinn_score + afinn_title_score
##
##                               Df Sum of Sq    RSS      AIC
## - afinn_title_score        1    0.42  580.56 -2643.40
## - afinn_score               1    0.42  580.56 -2643.39
## <none>                      580.14 -2642.91
```

```

## - CC           1     1.50  581.64 -2639.51
## - log(Length) 1     7.21  587.35 -2619.01
## - Category    21    376.47 956.61 -1635.65
## - log(Subscribers) 1   1410.26 1990.40   -58.45
##
## Step: AIC=-2643.4
## log(Views) ~ CC + log(Length) + log(Subscribers) + Category +
##      afinn_score
##
##                               Df Sum of Sq     RSS     AIC
## <none>                      580.56 -2643.40
## - afinn_score    1     0.62  581.18 -2643.15
## - CC            1     1.40  581.96 -2640.34
## - log(Length)   1     7.25  587.81 -2619.36
## - Category     21    376.15 956.71 -1637.43
## - log(Subscribers) 1   1415.80 1996.36   -54.18

summary(lm4)

```

```

##
## Call:
## lm(formula = log(Views) ~ CC + log(Length) + log(Subscribers) +
##      Category + afinn_score, data = df1)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -1.30364 -0.34696 -0.06379  0.29275  2.63096
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2.01453   0.10738 18.761 < 2e-16 ***
## CC1                       0.06063   0.02711  2.237 0.025411 *
## log(Length)                -0.08380  0.01647 -5.087 3.97e-07 ***
## log(Subscribers)           0.83081   0.01169 71.084 < 2e-16 ***
## CategoryAutomobile,Comedy  0.39759   0.07622  5.216 2.01e-07 ***
## CategoryBlog                 -0.21831  0.06273 -3.480 0.000512 ***
## CategoryBlog,Comedy          -0.13598  0.09120 -1.491 0.136110
## CategoryBlog,Entertainment   0.43821   0.14538  3.014 0.002608 **
## CategoryBlog,Science          -0.63493  0.16621 -3.820 0.000137 ***
## CategoryComedy                0.95713   0.10866  8.808 < 2e-16 ***
## CategoryComedy,Entertainment  0.83664   0.07187 11.641 < 2e-16 ***
## CategoryComedy,Informative    0.79422   0.08926  8.898 < 2e-16 ***
## CategoryEntertainment         0.39775   0.08487  4.686 2.96e-06 ***
## CategoryEntertainment,Blog    0.32099   0.11049  2.905 0.003710 **
## CategoryEntertainment,Comedy  0.90616   0.07233 12.528 < 2e-16 ***
## CategoryFood                  0.29563   0.05435  5.439 5.98e-08 ***
## CategoryFood,Entertainment    0.51036   0.11860  4.303 1.76e-05 ***
## CategoryInformative            0.04349   0.05775  0.753 0.451476
## CategoryNews                  0.20090   0.06208  3.236 0.001230 **
## CategoryScience                -0.04915  0.05430 -0.905 0.365534
## CategoryTech                  -0.58233  0.05878 -9.908 < 2e-16 ***
## CategoryTech,Comedy            -0.38563  0.12048 -3.201 0.001391 **
## CategoryTech,Informative       -1.79659  0.13119 -13.695 < 2e-16 ***
## CategoryTech,News              -1.01754  0.13520 -7.526 7.74e-14 ***

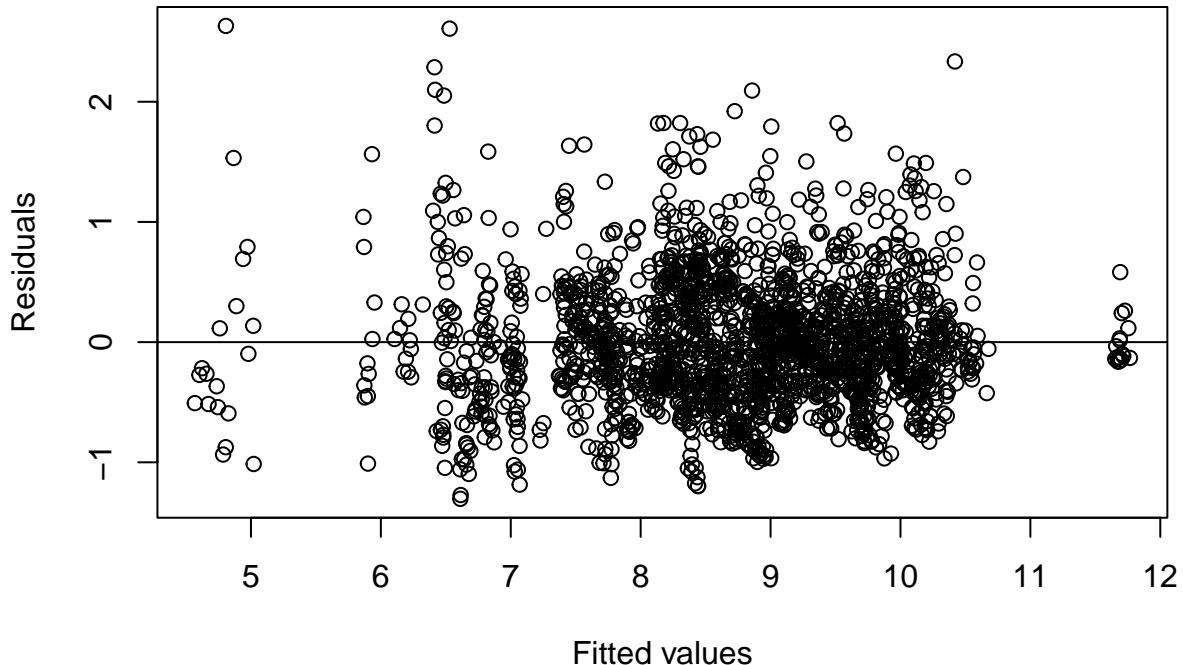
```

```

## CategoryVideoGames      -0.19716   0.05606  -3.517 0.000446 ***
## afinn_score            -0.06743   0.04527  -1.489 0.136523
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5293 on 2072 degrees of freedom
## Multiple R-squared:  0.8153, Adjusted R-squared:  0.813
## F-statistic: 365.8 on 25 and 2072 DF,  p-value: < 2.2e-16

plot(predict(lm4), resid(lm4), xlab = "Fitted values", ylab = "Residuals")
abline(h=0)

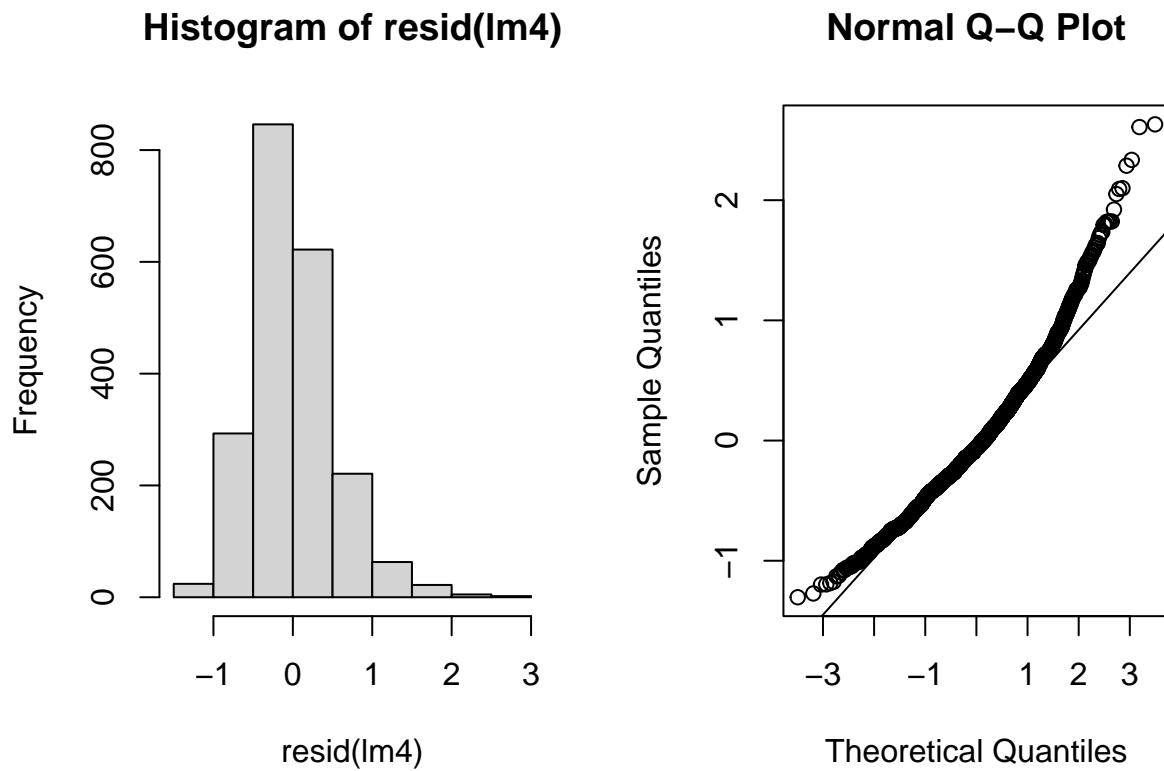
```



```

par(mfrow=c(1,2))
hist(resid(lm4))
qqnorm(resid(lm4))
qqline(resid(lm4))

```



```
# fit lm4 (the final model in MLR part) to df_train
lm5 <- lm(Views ~ CC + log(Length) + log(Subscribers) + Category + afinn_score, data = df_train)

# make prediction
pred1 <- predict(lm5, newdata = df_test)

# Compute the RMSE
RMSE(df_test$Views, pred1)

## [1] 8322.826
```

## Regression Tree

Fit a regression tree on the training set.

```
# Fit tree model
t1 <- rpart(Views ~ CC + Released + Category + Length + Subscribers + afinn_score + afinn_title_score,
            data = df_train,
            method = "anova")
summary(t1)

## Call:
## rpart(formula = Views ~ CC + Released + Category + Length + Subscribers +
```

```

##      afinn_score + afinn_title_score, data = df_train, method = "anova")
## n= 1469
##
##          CP nsplit rel error     xerror      xstd
## 1 0.31049310      0 1.0000000 1.0015040 0.2464760
## 2 0.11052780      1 0.6895069 0.7397027 0.1912329
## 3 0.06701585      2 0.5789791 0.6522043 0.1974818
## 4 0.04473830      3 0.5119633 0.6064179 0.1936102
## 5 0.03135581      4 0.4672250 0.5810376 0.1906906
## 6 0.01707528      5 0.4358692 0.5440516 0.1895579
## 7 0.01616416      6 0.4187939 0.5403960 0.1944559
## 8 0.01000000      7 0.4026297 0.5250192 0.1942670
##
## Variable importance
## Subscribers    Category      Length     Released afinn_score
##       63           25          6           4           3
##
## Node number 1: 1469 observations,      complexity param=0.3104931
##   mean=12259.92, MSE=3.485821e+08
##   left son=2 (1358 obs) right son=3 (111 obs)
## Primary splits:
##   Subscribers < 18200      to the left,  improve=0.310493100, (0 missing)
##   Category      splits as LLLLRLLLLLLRLRLRLRL, improve=0.223444300, (0 missing)
##   CC            splits as LR, improve=0.021839480, (0 missing)
##   Released      < 54      to the left,  improve=0.016626150, (0 missing)
##   afinn_score < -0.4679184 to the right, improve=0.005393465, (0 missing)
## Surrogate splits:
##   Category splits as LLLRLRLRLRLRLRLRL, agree=0.931, adj=0.09, (0 split)
##
## Node number 2: 1358 observations,      complexity param=0.06701585
##   mean=9285.58, MSE=9.976327e+07
##   left son=4 (877 obs) right son=5 (481 obs)
## Primary splits:
##   Subscribers < 5820      to the left,  improve=0.25329930, (0 missing)
##   Category      splits as LRLL-LRRRLRRLRRLRL, improve=0.13649260, (0 missing)
##   CC            splits as LR, improve=0.06061949, (0 missing)
##   Released      < 17.5     to the left,  improve=0.03278464, (0 missing)
##   afinn_score < -0.3582549 to the right, improve=0.01406778, (0 missing)
## Surrogate splits:
##   Category      splits as LLLL-LLLLLRLRLRLRL, agree=0.706, adj=0.170, (0 split)
##   afinn_score    < -0.4130772 to the right, agree=0.653, adj=0.021, (0 split)
##   afinn_title_score < -10.00973 to the right, agree=0.651, adj=0.015, (0 split)
##   Length         < 94      to the left,  agree=0.650, adj=0.012, (0 split)
##   Released       < 138     to the left,  agree=0.649, adj=0.010, (0 split)
##
## Node number 3: 111 observations,      complexity param=0.1105278
##   mean=48648.65, MSE=1.960318e+09
##   left son=6 (102 obs) right son=7 (9 obs)
## Primary splits:
##   Category      splits as ----R----L--L---L----L, improve=0.26010510, (0 missing)
##   Subscribers < 60800      to the left,  improve=0.26010510, (0 missing)
##   Released      < 30      to the right, improve=0.11589390, (0 missing)
##   Length         < 3.5     to the right, improve=0.07149841, (0 missing)
##   afinn_score < -0.09233514 to the left,  improve=0.04949090, (0 missing)

```

```

## Surrogate splits:
##   Subscribers < 60800      to the left, agree=1.000, adj=1.000, (0 split)
##   Released    < 10.5       to the right, agree=0.937, adj=0.222, (0 split)
##
## Node number 4: 877 observations, complexity param=0.01707528
##   mean=5562.734, MSE=3.63767e+07
##   left son=8 (529 obs) right son=9 (348 obs)
## Primary splits:
##   Subscribers < 2660      to the left, improve=0.27407660, (0 missing)
##   Category     splits as LLLL-LRRRR-RR-RRLLLLLL, improve=0.16847750, (0 missing)
##   CC           splits as LR, improve=0.04789304, (0 missing)
##   Released    < 17.5       to the left, improve=0.03350874, (0 missing)
##   Length       < 5.5       to the right, improve=0.02945479, (0 missing)
## Surrogate splits:
##   Category     splits as LLLL-LRRRL-LL-RRLLLLLL, agree=0.688, adj=0.213, (0 split)
##   afinn_score  < -0.4415035 to the right, agree=0.627, adj=0.060, (0 split)
##   CC           splits as LR, agree=0.620, adj=0.043, (0 split)
##   afinn_title_score < -3.477081 to the right, agree=0.613, adj=0.026, (0 split)
##
## Node number 5: 481 observations, complexity param=0.03135581
##   mean=16073.39, MSE=1.439907e+08
##   left son=10 (334 obs) right son=11 (147 obs)
## Primary splits:
##   Category     splits as LRLL---RR-LLRLLRL---L, improve=0.23182770, (0 missing)
##   Subscribers < 7815      to the left, improve=0.07282145, (0 missing)
##   Released    < 66         to the left, improve=0.06285942, (0 missing)
##   CC           splits as LR, improve=0.02763750, (0 missing)
##   Length       < 2.5       to the right, improve=0.01042117, (0 missing)
## Surrogate splits:
##   Subscribers < 16350      to the left, agree=0.771, adj=0.252, (0 split)
##   Released    < 78         to the left, agree=0.736, adj=0.136, (0 split)
##
## Node number 6: 102 observations, complexity param=0.0447383
##   mean=41941.18, MSE=1.47584e+09
##   left son=12 (89 obs) right son=13 (13 obs)
## Primary splits:
##   Length       < 3.5       to the right, improve=0.15218330, (0 missing)
##   Subscribers < 29950      to the left, improve=0.07016891, (0 missing)
##   Category     splits as -----L--L---R---L, improve=0.04652797, (0 missing)
##   afinn_score  < 0.2123762 to the right, improve=0.02369423, (0 missing)
##   afinn_title_score < 0.1749704 to the right, improve=0.01726483, (0 missing)
## Surrogate splits:
##   afinn_score < -0.6155792 to the right, agree=0.931, adj=0.462, (0 split)
##
## Node number 7: 9 observations
##   mean=124666.7, MSE=1.162444e+09
##
## Node number 8: 529 observations
##   mean=3001.735, MSE=8899958
##
## Node number 9: 348 observations
##   mean=9455.747, MSE=5.301896e+07
##
## Node number 10: 334 observations

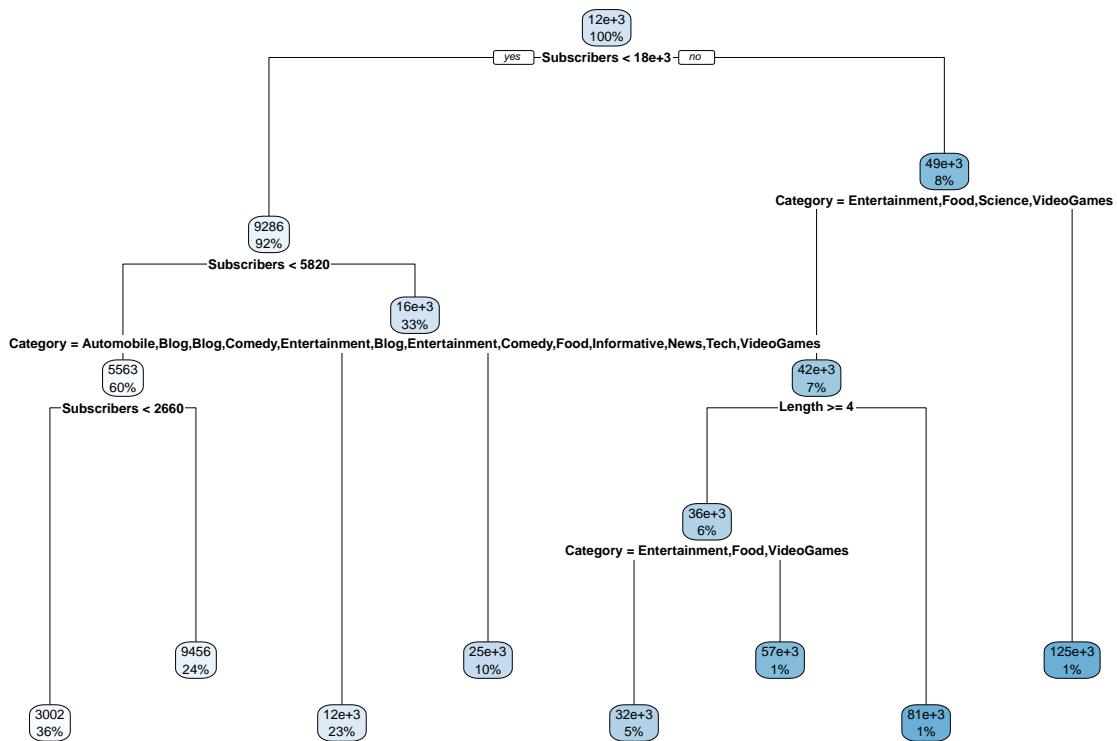
```

```

##   mean=12240.42, MSE=5.324768e+07
##
## Node number 11: 147 observations
##   mean=24782.31, MSE=2.409424e+08
##
## Node number 12: 89 observations,    complexity param=0.01616416
##   mean=36213.48, MSE=3.726848e+08
##   left son=24 (73 obs) right son=25 (16 obs)
## Primary splits:
##   Category      splits as  -----L--L---R----L, improve=0.24954480, (0 missing)
##   Released      < 30          to the right, improve=0.09665358, (0 missing)
##   Subscribers   < 24550       to the right, improve=0.08817957, (0 missing)
##   afinn_score   < -0.09233514 to the left,  improve=0.06772828, (0 missing)
##   Length        < 11.5        to the right, improve=0.04553668, (0 missing)
## Surrogate splits:
##   Released < 17.5          to the right, agree=0.843, adj=0.125, (0 split)
##
## Node number 13: 13 observations
##   mean=81153.85, MSE=7.265976e+09
##
## Node number 24: 73 observations
##   mean=31698.63, MSE=1.994434e+08
##
## Node number 25: 16 observations
##   mean=56812.5, MSE=6.457773e+08

# Plot the desicion tree
rpart.plot(t1)

```



```

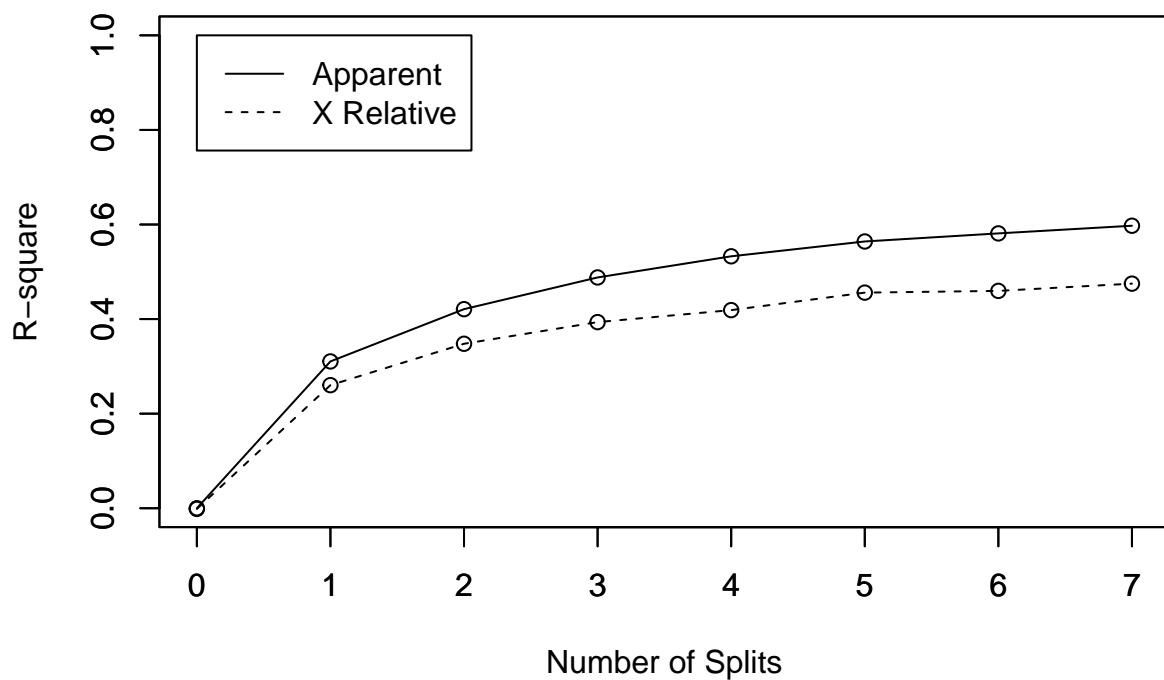
# Plot R-square vs Splits and the Relative Error vs Splits.
rsq.rpart(t1)

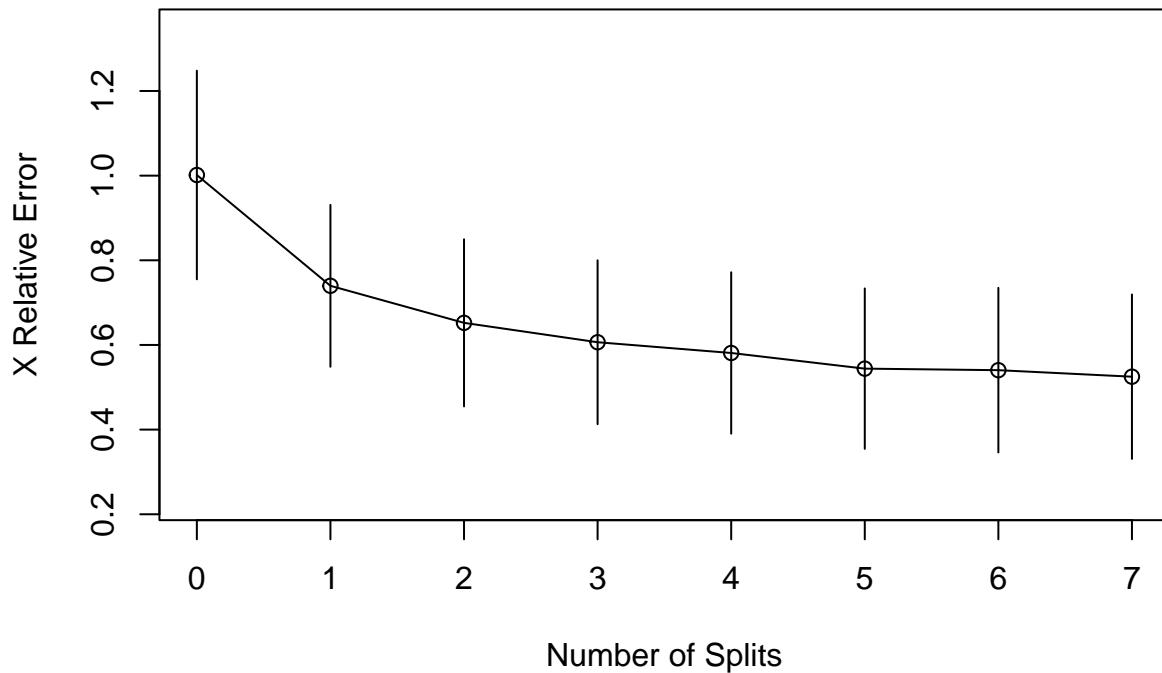
```

```

## 
## Regression tree:
## rpart(formula = Views ~ CC + Released + Category + Length + Subscribers +
##       afinn_score + afinn_title_score, data = df_train, method = "anova")
##
## Variables actually used in tree construction:
## [1] Category      Length      Subscribers
##
## Root node error: 5.1207e+11/1469 = 348582113
##
## n= 1469
##
##          CP nsplit rel error  xerror     xstd
## 1 0.310493      0  1.00000 1.00150 0.24648
## 2 0.110528      1  0.68951 0.73970 0.19123
## 3 0.067016      2  0.57898 0.65220 0.19748
## 4 0.044738      3  0.51196 0.60642 0.19361
## 5 0.031356      4  0.46722 0.58104 0.19069
## 6 0.017075      5  0.43587 0.54405 0.18956
## 7 0.016164      6  0.41879 0.54040 0.19446
## 8 0.010000      7  0.40263 0.52502 0.19427

```





Make predictions on the test set and compute the RMSE

```
# Make prediction
pred_tree <- predict(t1, newdata = df_test)

# Compute the RMSE
RMSE(df_test$Views, pred_tree)

## [1] 7977.73

# test R^2
cor(df_test$Views, pred_tree)^2

## [1] 0.7801641
```

## Random Forest

Fit a Random Forest on the training set usinng the defaults for mtry and ntree.

```
set.seed(652)
rf1 <- randomForest(Views ~ CC + Released + Category + Length + Subscribers + afinn_score + afinn_title,
```

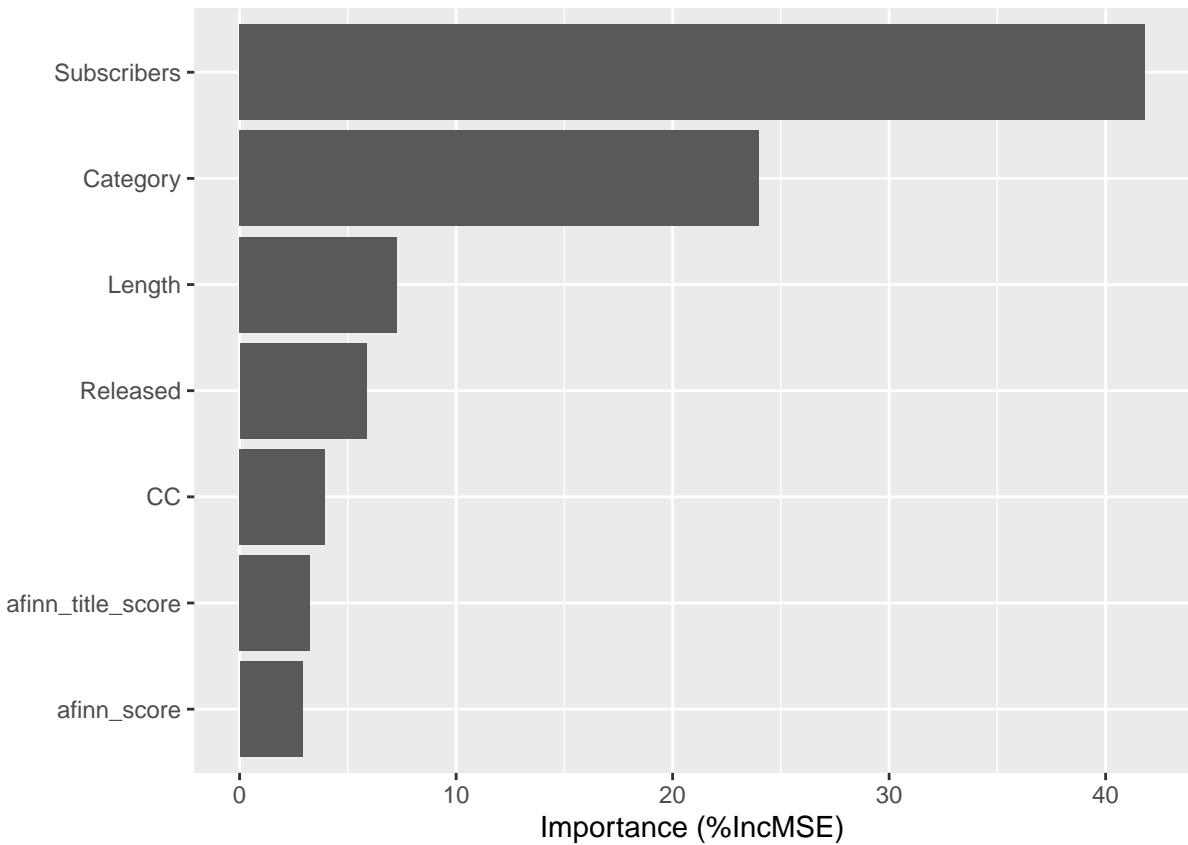
```

## 
## Call:
##   randomForest(formula = Views ~ CC + Released + Category + Length +      Subscribers + afinn_score +
##                 Type of random forest: regression
##                 Number of trees: 500
##   No. of variables tried at each split: 2
##
##   Mean of squared residuals: 154686582
##   % Var explained: 55.62

```

Use the `vip()` function to make a variable importance plot.

```
vip(rf1, num_features = 14, include_type = TRUE)
```



Make predictions on the test set and compute the RMSE

```

# Make prediction
pred_rf <- predict(rf1, newdata = df_test)

# Compute the RMSE
RMSE(df_test$Views, pred_rf)

```

```
## [1] 6660.282
```

```
# test R^2
cor(df_test$Views, pred_rf)^2
```

```
## [1] 0.836752
```

## Conclusion(TODO)

Model	RMSE	R Squared	Number of Coefficients	performance	interpretability
linear regression	-	-	-	-	-
regression tree	-	-	-	-	-
random forest	-	-	-	-	-

## Linear Regression vs Regression Tree VS Random Forest

Aggregated/ensemble models are not universally better than their “single” counterparts, they are better if and only if the single models suffer of instability. With XX training rows and only XX columns, we are in a comfortable training sample size situation in which even a decision tree may get reasonably stable.