Project - Online News Popularity

Code

Hide

```
rm(list = ls())
```

In this study, we need the following packages

Hide

```
library(tidyverse)
library(glmnet)
library(leaps)
library(randomForest)
theme_set(theme_bw())
```

1. Description of the pratical problem

Mashable is a global, multi-platform media and entertainment company. The following dataset summarizes a heterogeneous set of features about articles published by Mashable in a period of two years. The database can be found at the folowing address: https://archive.ics.uci.edu/ml/datasets/online+news+popularity (https://archive.ics.uci.edu/ml/datasets/online+news+popularity)

We read the data.

Hide

```
mydata = read.csv(file = "OnlineNewsPopularity.csv", header=TRUE, sep=",", na.strings = "NA")
head(mydata, n = 1)
```

	url <fctr>	timedelta <dbl>	n_tokens_title <dbl>
1	http://mashable.com/2013/01/07/amazon-instant-video-browser/	731	12

1 row | 1-4 of 61 columns

Hide

```
dim(mydata)
```

```
[1] 39644    61
```

Our data set is quite large. We have 39644 observations and 61 variables. The goal is to predict the number of shares on social networks (popularity).

2. Description of the mathematical problem

The problem is to explain the output that will be named “nb_shares” by the other variables of the dataset (we will exclude some of them). We denote by Y the dependent variable “nb_shares” and by X_n the n independent variables. We want to explain the connection between Y and $X_1, \dots, X_n$.

Y being continuous, we are facing a regression problem. We want to find a machine that will minimize the error between one prediction of our model y' and one observation Y. In order to measure this error, we will use the **quadratic risk function** for the machine m that is $R(m) = E[(Y - m(X))^2]$ to try to find a machine m such as $m_n(x) \approx m^\star(x)$.

As we will see later, we will estimate these risks by splitting our dataset into a **train** dataset and **test** dataset.

3. Preparation and cleaning of the data

Cleaning of the names of the columns

- **Transform lower cases**

Hide

```
colnames(mydata) = tolower(colnames(mydata))
```

- **Change the columns names to have more explicit names**

Hide

```
colnames(mydata)[colnames(mydata)=="n_tokens_title"] = "nb_words_title"
colnames(mydata)[colnames(mydata)=="n_tokens_content"] = "nb_words_content"
colnames(mydata)[colnames(mydata)=="n_unique_tokens"] = "rate_unique_words"
colnames(mydata)[colnames(mydata)=="n_non_stop_words"] = "rate_non_stop_words"
colnames(mydata)[colnames(mydata)=="n_non_stop_unique_tokens"] = "rate_non_stop_unique_words"
colnames(mydata)[colnames(mydata)=="num_hrefs"] = "nb_links"
# Number of links to other articles published by Mashable
colnames(mydata)[colnames(mydata)=="num_self_hrefs"] = "nb_links_mashable"
colnames(mydata)[colnames(mydata)=="num_imgs"] = "nb_images"
colnames(mydata)[colnames(mydata)=="num_videos"] = "nb_videos"
colnames(mydata)[colnames(mydata)=="average_token_length"] = "average_word_length"
colnames(mydata)[colnames(mydata)=="num_keywords"] = "nb_keywords"
colnames(mydata)[colnames(mydata)=="shares"] = "nb_shares"  # Target variable
```

- **Cleaning the data**
  We have ratio superior to 1. It is not possible. We find the row where the ratio of rate_non_stop_words is maximum.

Hide

```
index_row = which.max(mydata$rate_non_stop_words)
mydata[index_row,]
```

	url <fctr>	timedelta <dbl>	nb_words_tit <db
31038	http://mashable.com/2014/08/18/ukraine-civilian-convoy-attacked/	142	

1 row | 1-4 of 61 columns

A lot of aberrant values. We drop this row.

Hide

```
mydata = mydata[-index_row,]
```

We also detect an article with 0 words. We check it.

Hide

```
index_row = which.min(mydata$nb_words_content)
mydata[index_row,]
```

	url <fctr>	timedelta <dbl>	nb_words_title <dbl>
894	http://mashable.com/2013/01/23/actual-facebook-graph-searches/	715	10

1 row | 1-4 of 61 columns

This is a mistake there are words in this article. We exclude the rows without words.

Hide

```
dim(mydata) # Before exclusion
```

```
[1] 39643    61
```

Hide

```
mydata = mydata %>% filter(nb_words_content!=0)
dim(mydata) # After exclusion
```

```
[1] 38462    61
```

We drop the column rate_non_stop_words with only 1 in values.

Hide

```
mydata = mydata %>% select(-rate_non_stop_words)
```

- **NA values**
  We check that we do not have NA values. Look at the number by column.

Hide

```
na_columns = sapply(mydata, function(x) sum(is.na(x)))
na_columns[na_columns>0]
```

```
named integer(0)
```

We do not have any missing values.

- **Creation of categorical variables**
  0 - Create an ID by row

Hide

```
number_rows = dim(mydata)[1]
ID = data.frame(ID = c(1:number_rows))
mydata = cbind(ID,mydata)
```

Transforming the categorical variables to factor.

Hide

```
columns_weekdays = mydata %>% select(contains("weekday")) %>% colnames
columns_channel = mydata %>% select(contains("data_channel_is")) %>% colnames
other_columns = c("is_weekend")
# All the columns names in one vector
columns_to_factor = c(columns_weekdays,columns_channel,other_columns)
# We change them to factor.
mydata[columns_to_factor] = lapply(mydata[columns_to_factor], factor)
```

- **Group the days and the channel in two variables.**
  We have 7 categorical variables: weekday_is_monday and so on. We create a new categorical variable “weekday” with the name of the day.

Hide

```
df_weekday = mydata %>% select(ID, columns_weekdays) %>% gather(key=weekday, value=value_to_filter, -I
D)
df_weekday = df_weekday %>% mutate(weekday=recode(weekday,
                                                                        "weekday_is_monday"="monday",
                                                                        "weekday_is_tuesday"="tuesda
y",
                                                                        "weekday_is_wednesday"="wedne
sday",
                                                                        "weekday_is_thursday"="thursd
ay",
                                                                        "weekday_is_friday"="friday",
                                                                        "weekday_is_saturday"="saturd
ay",
                                                                        "weekday_is_sunday"="sunday"
                                                                        ))
df_weekday = df_weekday %>% filter(value_to_filter==1)
df_weekday$weekday = as.factor(df_weekday$weekday)
df_weekday$weekday = ordered(df_weekday$weekday, levels=c("monday","tuesday","wednesday","thursday","f
riday","saturday","sunday"))
df_weekday = df_weekday %>% select(ID, weekday)
mydata = left_join(mydata, df_weekday,by="ID")
```

We have 6 categorical variables: data_channel_is_lifestyle and so on. We create a new categorical variable “data_channel” with the name of the channel.

Hide

```
df_channel = mydata %>% select(ID, columns_channel) %>% gather(key=channel, value=value_to_filter, -I
D)
df_channel = df_channel %>% mutate(channel=recode(channel,
                                                                     "data_channel_is_lifestyle"=
"lifestyle",
                                                                     "data_channel_is_entertainmen
t"="entertainment",
                                                                     "data_channel_is_bus"="bus",
                                                                     "data_channel_is_socmed"="soc
med",
                                                                     "data_channel_is_tech"="tech"
,
                                                                     "data_channel_is_world"="worl
d"
                                                                     ))
df_channel = df_channel %>% filter(value_to_filter==1)
df_channel$channel = as.factor(df_channel$channel)
levels(df_channel$channel) = c(levels(df_channel$channel),"other")
df_channel = df_channel %>% select(ID, channel)
mydata = left_join(mydata, df_channel,by="ID")
mydata$channel[is.na(mydata$channel)] = "other"
```

- **Removal of non predictive variables**
  We move the target variable to first position after the non predictive variables.

Hide

```
mydata = mydata %>% select(url, ID, timedelta, nb_shares, everything())
```

To create the models, we create a data.frame “mydata2” without the doublons and the non predictive variables.

Hide

```
mydata2 = mydata %>% select(-c(columns_channel, columns_weekdays,url, ID, timedelta))
```

4. Description of the dataset (with dplyr)

Hide

```
summary(mydata2)
```

```
   nb_shares      nb_words_title  nb_words_content rate_unique_words rate_non_stop_unique_words    nb_
links      nb_links_mashable
 Min.   :     1   Min.   : 2.00   Min.   :  18.0   Min.   :0.1150    Min.   :0.1191             Min.
:  0.00   Min.   :  0.000
 1st Qu.:   945   1st Qu.: 9.00   1st Qu.: 259.0   1st Qu.:0.4774    1st Qu.:0.6326             1st Q
u.:  5.00   1st Qu.:  1.000
 Median :  1400   Median :10.00   Median : 423.0   Median :0.5430    Median :0.6937             Median
:  8.00   Median :  3.000
 Mean   :  3355   Mean   :10.38   Mean   : 563.3   Mean   :0.5468    Mean   :0.6935             Mean
: 11.22   Mean   :  3.395
 3rd Qu.:  2700   3rd Qu.:12.00   3rd Qu.: 729.0   3rd Qu.:0.6111    3rd Qu.:0.7569             3rd Q
u.: 14.00   3rd Qu.:  4.000
 Max.   :843300   Max.   :23.00   Max.   :8474.0   Max.   :1.0000    Max.   :1.0000             Max.
:304.00   Max.   :116.000


   nb_images         nb_videos       average_word_length  nb_keywords       kw_min_min       kw_max_min
      kw_avg_min
 Min.   :  0.000   Min.   : 0.000   Min.   :3.600       Min.   : 1.000   Min.   : -1.00   Min.   :
0   Min.   :   -1.0
 1st Qu.:  1.000   1st Qu.: 0.000   1st Qu.:4.496       1st Qu.: 6.000   1st Qu.: -1.00   1st Qu.:   4
45   1st Qu.:  143.0
 Median :  1.000   Median : 0.000   Median :4.674       Median : 7.000   Median : -1.00   Median :   6
60   Median :  237.6
 Mean   :  4.562   Mean   : 1.264   Mean   :4.688       Mean   : 7.215   Mean   : 26.71   Mean   :  11
52   Mean   :  313.9
 3rd Qu.:  4.000   3rd Qu.: 1.000   3rd Qu.:4.862       3rd Qu.: 9.000   3rd Qu.:  4.00   3rd Qu.:  10
00   3rd Qu.:  359.2
 Max.   :128.000   Max.   :91.000   Max.   :8.042       Max.   :10.000   Max.   :377.00   Max.   :2984
00   Max.   :42827.9


   kw_min_max        kw_max_max        kw_avg_max        kw_min_avg     kw_max_avg        kw_avg_avg     se
lf_reference_min_shares
 Min.   :     0   Min.   :     0   Min.   :     0   Min.   :  -1   Min.   :     0   Min.   :    0   Mi
n.   :     0
 1st Qu.:     0   1st Qu.:843300   1st Qu.:171300   1st Qu.:    0   1st Qu.:  3549   1st Qu.: 2374   1s
t Qu.:   703
 Median :  1400   Median :843300   Median :242080   Median :1009   Median :  4312   Median : 2851   Me
dian :  1200
 Mean   : 13182   Mean   :750315   Mean   :255213   Mean   :1102   Mean   :  5604   Mean   : 3103   Me
an   :  4122
 3rd Qu.:  7700   3rd Qu.:843300   3rd Qu.:326864   3rd Qu.:2031   3rd Qu.:  5962   3rd Qu.: 3551   3r
d Qu.:  2700
 Max.   :843300   Max.   :843300   Max.   :843300   Max.   :3613   Max.   :298400   Max.   :43568   Ma
x.   :843300


 self_reference_max_shares self_reference_avg_sharess is_weekend     lda_00            lda_01
   lda_02
 Min.   :     0            Min.   :     0             0:33436    Min.   :0.01818   Min.   :0.01818   M
in.   :0.01818
 1st Qu.:  1200            1st Qu.:  1100             1: 5026    1st Qu.:0.02506   1st Qu.:0.02501   1
st Qu.:0.02857
 Median :  3000            Median :  2300                        Median :0.03342   Median :0.03335   M
edian :0.04001
 Mean   : 10647            Mean   :  6598                        Mean   :0.18814   Mean   :0.14168   M
ean   :0.21718
 3rd Qu.:  8200            3rd Qu.:  5300                        3rd Qu.:0.25195   3rd Qu.:0.15069   3
rd Qu.:0.33502
 Max.   :843300            Max.   :843300                        Max.   :0.92699   Max.   :0.92595   M
ax.   :0.92000


     lda_03            lda_04         global_subjectivity global_sentiment_polarity global_rate_positiv
e_words
 Min.   :0.01818   Min.   :0.01818   Min.   :0.0000      Min.   :-0.39375          Min.   :0.00000

 1st Qu.:0.02562   1st Qu.:0.02858   1st Qu.:0.4025      1st Qu.: 0.06439          1st Qu.:0.02947
```

```
 Median :0.04000   Median :0.05000   Median :0.4566      Median : 0.12252          Median :0.03960

 Mean   :0.21430   Mean   :0.23870   Mean   :0.4570      Mean   : 0.12298          Mean   :0.04084

 3rd Qu.:0.34052   3rd Qu.:0.41461   3rd Qu.:0.5103      3rd Qu.: 0.17992          3rd Qu.:0.05072

 Max.   :0.92653   Max.   :0.92719   Max.   :1.0000      Max.   : 0.72784          Max.   :0.15549



 global_rate_negative_words rate_positive_words rate_negative_words avg_positive_polarity min_positive
_polarity
 Min.   :0.00000            Min.   :0.0000      Min.   :0.0000      Min.   :0.0000        Min.   :0.00
000
 1st Qu.:0.01018            1st Qu.:0.6129      1st Qu.:0.2000      1st Qu.:0.3119        1st Qu.:0.05
000
 Median :0.01567            Median :0.7143      Median :0.2857      Median :0.3619        Median :0.10
000
 Mean   :0.01712            Mean   :0.7031      Mean   :0.2968      Mean   :0.3647        Mean   :0.09
838
 3rd Qu.:0.02199            3rd Qu.:0.8000      3rd Qu.:0.3871      3rd Qu.:0.4133        3rd Qu.:0.10
000
 Max.   :0.18493            Max.   :1.0000      Max.   :1.0000      Max.   :1.0000        Max.   :1.00
000


 max_positive_polarity avg_negative_polarity min_negative_polarity max_negative_polarity title_subject
ivity
 Min.   :0.00          Min.   :-1.0000       Min.   :-1.0000       Min.   :-1.0000       Min.   :0.000
0
 1st Qu.:0.60          1st Qu.:-0.3315       1st Qu.:-0.7143       1st Qu.:-0.1250       1st Qu.:0.000
0
 Median :0.80          Median :-0.2577       Median :-0.5000       Median :-0.1000       Median :0.125
0
 Mean   :0.78          Mean   :-0.2675       Mean   :-0.5380       Mean   :-0.1108       Mean   :0.280
6
 3rd Qu.:1.00          3rd Qu.:-0.1934       3rd Qu.:-0.3125       3rd Qu.:-0.0500       3rd Qu.:0.500
0
 Max.   :1.00          Max.   : 0.0000       Max.   : 0.0000       Max.   : 0.0000       Max.   :1.000
0


 title_sentiment_polarity abs_title_subjectivity abs_title_sentiment_polarity      weekday
  channel
 Min.   :-1.0000          Min.   :0.0000         Min.   :0.0000                 monday   :6471   bus
      :6235
 1st Qu.: 0.0000          1st Qu.:0.1667         1st Qu.:0.0000                 tuesday  :7170   enterta
inment:6855
 Median : 0.0000          Median :0.5000         Median :0.0000                 wednesday:7205   lifesty
le    :2077
 Mean   : 0.0710          Mean   :0.3424         Mean   :0.1549                 thursday :7052   socmed
      :2311
 3rd Qu.: 0.1364          3rd Qu.:0.5000         3rd Qu.:0.2500                 friday   :5538   tech
      :7325
 Max.   : 1.0000          Max.   :0.5000         Max.   :1.0000                 saturday :2369   world
      :8168
                                                                                sunday   :2657   other

      :5491
```

Hide

```
colnames(mydata2)
```

```
 [1] "nb_shares"                     "nb_words_title"               "nb_words_content"              "rat
e_unique_words"
 [5] "rate_non_stop_unique_words"    "nb_links"                     "nb_links_mashable"             "nb_
images"
 [9] "nb_videos"                     "average_word_length"          "nb_keywords"                   "kw_
min_min"
[13] "kw_max_min"                    "kw_avg_min"                   "kw_min_max"                    "kw_
max_max"
[17] "kw_avg_max"                    "kw_min_avg"                   "kw_max_avg"                    "kw_
avg_avg"
[21] "self_reference_min_shares"     "self_reference_max_shares"    "self_reference_avg_sharess"    "is_
weekend"
[25] "lda_00"                        "lda_01"                       "lda_02"                        "lda
_03"
[29] "lda_04"                        "global_subjectivity"          "global_sentiment_polarity"     "glo
bal_rate_positive_words"
[33] "global_rate_negative_words"    "rate_positive_words"          "rate_negative_words"           "avg
_positive_polarity"
[37] "min_positive_polarity"         "max_positive_polarity"        "avg_negative_polarity"         "min
_negative_polarity"
[41] "max_negative_polarity"         "title_subjectivity"           "title_sentiment_polarity"      "abs
_title_subjectivity"
[45] "abs_title_sentiment_polarity"  "weekday"                      "channel"
```

Columns numeric

Hide

```
columns_numeric <- unlist(lapply(mydata2, is.numeric))
```

- **We follow our intuition and look at some correlations that could be interesting**

Hide

```
cor(mydata2[,c("nb_shares","nb_words_title","nb_words_content","nb_images","nb_videos")])
```

```
                   nb_shares nb_words_title nb_words_content    nb_images   nb_videos
nb_shares        1.000000000    0.006212729      0.006701789  0.041279165  0.02471476
nb_words_title   0.006212729    1.000000000      0.028162440 -0.006525119  0.05246549
nb_words_content 0.006701789    0.028162440      1.000000000  0.352948886  0.10205617
nb_images        0.041279165   -0.006525119      0.352948886  1.000000000 -0.06657575
nb_videos        0.024714759    0.052465492      0.102056168 -0.066575748  1.00000000
```

The correlations are lower that we expected ! Having a lot of content in an article does not make it always popular.

- **Overview - Main features of an article**

Hide

```
mydata2%>%select(Length.title=nb_words_title,Nb_Links=nb_links,Subjectivity=global_subjectivity,Positi
ve=max_positive_polarity,Content=nb_words_content)%>%summarise_all(funs(mean))
```

Length.title	Nb_Links	Subjectivity	Positive	Content
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
10.38246	11.21788	0.4569957	0.779983	563.2692

1 row

- **Does adding multimedia content have an impact of number of shares?**

Hide

```
mydata2%>%select(nb_images,nb_videos,nb_shares)%>%mutate(multimedia_content=nb_images+nb_videos)%>%gro
up_by(multimedia_content)%>%summarise(mean_shares=mean(nb_shares))%>%arrange(desc(mean_shares))%>%filt
er(mean_shares<10000)%>%head()
```

multimedia_content	mean_shares
<dbl>	<dbl>

multimedia_content	mean_shares
<dbl>	<dbl>
55	9588.375
71	9437.000
90	9400.000
46	8450.000
60	8243.667
58	7500.000

6 rows

Articles with the higher shares mean were article with lots of multimedia content (excluding articles considered as outliers with more than 10 000 shares).

- **Does an article with a subjective title drive more shares?**

Hide

```
subjectivity=mydata2%>%select("nb_shares","title_subjectivity")%>%filter(nb_shares<20000)
subjectivity$title_subjectivity=as.factor(as.numeric(subjectivity$title_subjectivity>0.75))
subjectivity%>%select(title_subjectivity,nb_shares)%>%group_by(title_subjectivity)%>%summarise(Shares=
mean(nb_shares))%>%arrange(desc(Shares))
```

title_subjectivity	Shares
<fctr>	<dbl>
1	2838.307
0	2379.921

2 rows

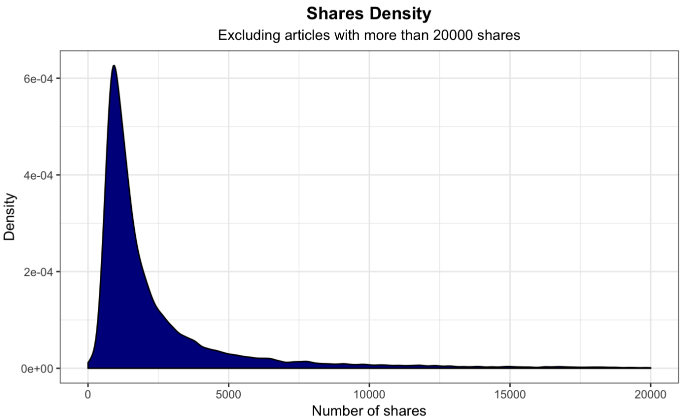
An article with subjectivity in its title tends to drive slightly more shares in average.

5. Vizualization of the dataset (with ggplot)

- **How the number of shares is distributed?**

Hide

```
ggplot(mydata2) + aes(x=nb_shares) + geom_density(fill="darkblue") + xlim(0,20000)+xlab("Number of sha
res")+ylab("Density")+ labs(title = "Shares Density", subtitle = "Excluding articles with more than 20
000 shares") +
  theme(plot.title = element_text(hjust = 0.5,face="bold"),plot.subtitle = element_text(hjust = 0.5))
```

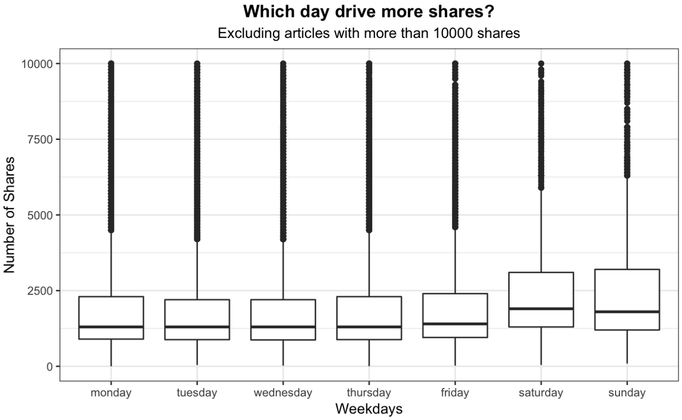


We observe that most of the articles have a number of shares less than 5000.

- **Which weekday drive more shares?**

Hide

```
ggplot(mydata2) + aes(x=weekday, y=nb_shares) + geom_boxplot() + ylim(0, 10000)+xlab("Weekdays")+ylab(
"Number of Shares")+ labs(title = "Which day drive more shares?", subtitle = "Excluding articles with
 more than 10000 shares") +
  theme(plot.title = element_text(hjust = 0.5,face="bold"),plot.subtitle = element_text(hjust = 0.5))
```

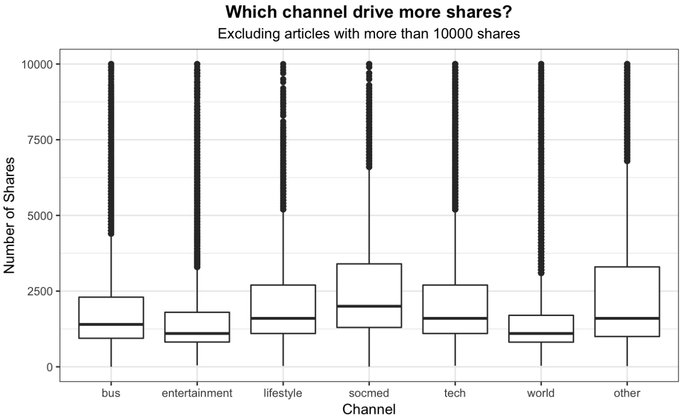


We can observe that Saturday and Sunday drive more shares in average than the other weekdays.

- **Which channel drive more shares?**

Hide

```
ggplot(mydata2) + aes(x=channel, y=nb_shares) + geom_boxplot() + ylim(0, 10000)+xlab("Channel")+ylab(
"Number of Shares")+ labs(title = "Which channel drive more shares?", subtitle = "Excluding articles w
ith more than 10000 shares") +
  theme(plot.title = element_text(hjust = 0.5,face="bold"),plot.subtitle = element_text(hjust = 0.5))
```

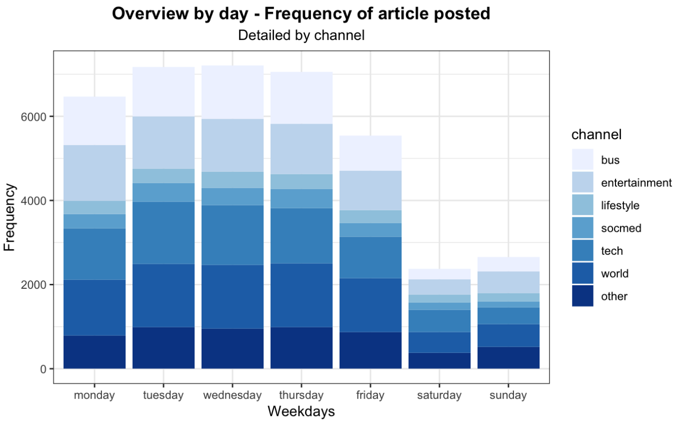


The above boxplots show that lifestyle, socmed, tech and other are the four channels that drive in average more shares as we have seen in the description of data.

- **Comparison between weekday & Channel**

Hide

```
ggplot(mydata2) + aes(x=weekday,fill=channel)+ geom_bar(position = "stack")+scale_fill_brewer(palette
 = "Blues")+xlab("Weekdays")+ylab("Frequency")+ labs(title = "Overview by day - Frequency of article p
osted", subtitle = "Detailed by channel") +
  theme(plot.title = element_text(hjust = 0.5,face="bold"),plot.subtitle = element_text(hjust = 0.5))
```

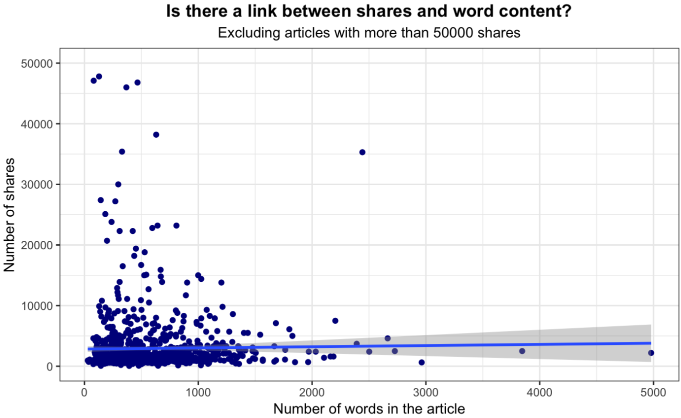


We mentionned before that during weekend, articles that are posted drive more shares. Even though, we notice that there is less articles posted compared to other weekdays.

- **Is there a link between shares and word content?**
  To do this plot we select a sub-sample.

Hide

```
mydata2_temp = sample_n(tbl = mydata2, size = 1000)
ggplot(mydata2_temp) + aes(x=nb_words_content,y=nb_shares)+ geom_point(color="darkblue") +
  geom_smooth(method="lm") +
  ylim(0,50000)+ylab("Number of shares")+xlab("Number of words in the article") +
  labs(title = "Is there a link between shares and word content?", subtitle = "Excluding articles with
 more than 50000 shares") +
  theme(plot.title = element_text(hjust = 0.5,face="bold"),plot.subtitle = element_text(hjust = 0.5))
```



The above graph suggests that the linear relation is week. However we can see that most of the very shared articles do not have a lot of words.

6. Machine Learning Methods

We propose to use the following methods to predict the number of shares:

1 - Linear model
2 - Penalized Regression (Lasso and Ridge)
3 - Random Forest

We create a copy of that dataset that will be used for models

Hide

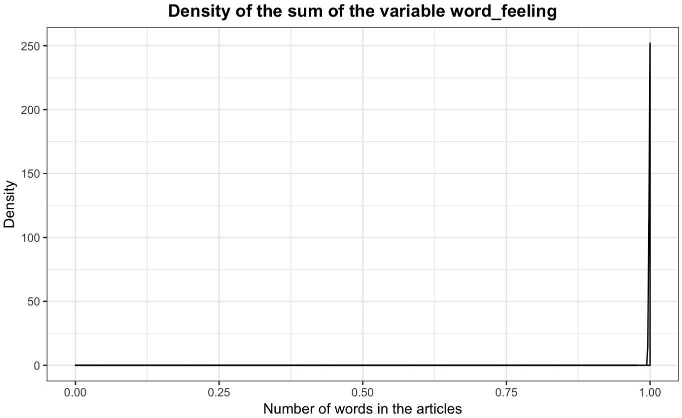
```
data <- mydata2
```

Additional preliminary cleaning

There are linear dependencies in the data that will prevent us from running some models :
rate_positive_words + rate_negative_words = 1

Hide

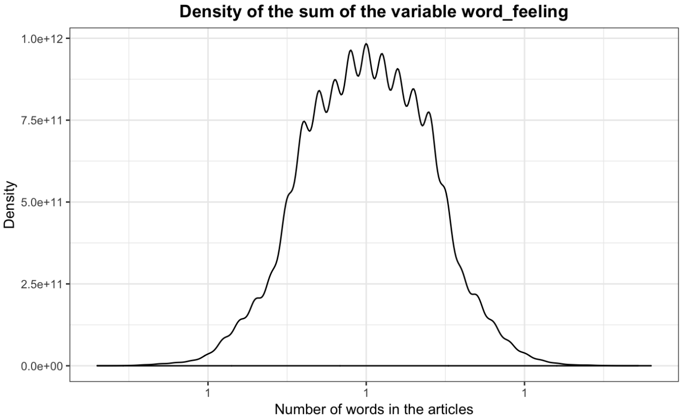
```
data %>% mutate(word_feeling = rate_positive_words + rate_negative_words) %>% ggplot() + aes(x = word_
feeling) + geom_density() + ylab("Density")+xlab("Number of words in the articles") +
  labs(title = "Density of the sum of the variable word_feeling") +  theme(plot.title = element_text(h
just = 0.5,face="bold"))
```



lda_00 + lda_01 + lda_02 + lda_03 + lda_04 = 1

Hide

```
data %>% mutate(sum_lda = lda_00 + lda_01 + lda_02 + lda_03 + lda_04) %>% ggplot() + aes(x = sum_lda)
+ geom_density() + ylab("Density")+xlab("Number of words in the articles") +
  labs(title = "Density of the sum of the variable word_feeling") +  theme(plot.title = element_text(h
just = 0.5,face="bold"))
```



Hide

```
summary(data$lda_00 + data$lda_01 + data$lda_02 + data$lda_03 + data$lda_04)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      1       1       1       1       1       1 
```

In absciss, we have values of 1 due to error approximation in the sum. If we had truncated the values, it would have been only a peak at 1.
We exclude the two problematic variables.

Hide

```
data <- data %>% select(-c(lda_04,rate_negative_words))
```

Linear Model

We will try to predict the number of shares based on all other regressors We start naively by trying to fit a Linear Regression to the whole dataset

Hide

```
# model.lm <- lm(nb_shares~.,data=data)
# Error: cannot allocate vector of size 11.0 Gb
```

R cannot compute it, due to the excess of parameters We will therefore : - work only on a sample of the dataset - consider numerical values only

Hide

```
Name_numeric_colums <- rownames(data.frame(columns_numeric[columns_numeric]))
Name_numeric_colums <- Name_numeric_colums[Name_numeric_colums != "lda_04" & Name_numeric_colums != "r
ate_negative_words"]

set.seed(123456)
data <- data %>% sample_n(10000) %>% select(Name_numeric_colums)
```

We can now separate in training and testing datasets

Hide

```
data.train <- data %>% slice(1:8000)
data.test <- data %>% slice(8001:10000)
```

We can now run a naive prediction model (that predicts the sample mean of the target column in the train data) and a linear model

Hide

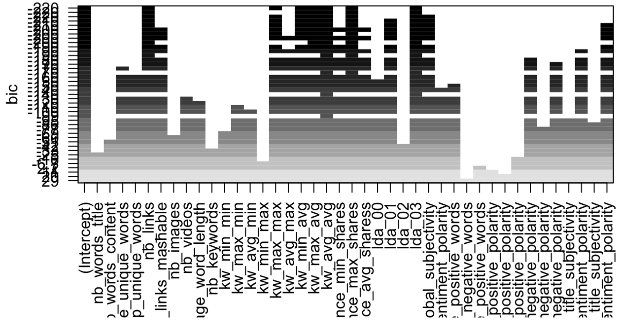
```
pred.naive <- rep(mean(data.train$nb_shares),nrow(data.test))
model.lm <- lm(nb_shares~.,data=data.train)
```

We do a subset selection. We select the **BIC** criteria rather than the **AIC** criteria because it penalises more aggressively the large models. Indeed, compared to **AIC**, **BIC** increases by $log(n)d$ whereas **AIC** increases by $2d$. For n = 10000, we will have $log(n)d \approx 9.21$ which is superior to 2 of the **AIC**.

Hide

```
reg.fit.forward <- regsubsets(nb_shares~.,data=data.train,method="forward",nvmax=150)
reg.fit.backward <- regsubsets(nb_shares~.,data=data.train,method="backward",nvmax=150)
sum.reg.fit.f <- summary(reg.fit.forward)
sum.reg.fit.b <- summary(reg.fit.backward)
nb.bic.f <- order(sum.reg.fit.f$bic)[1]
nb.bic.b <- order(sum.reg.fit.b$bic)[1]
get.formula<- function(a,number){
var.sel <- a$which[number,][-1]
var.sel1 <- names(var.sel)[var.sel] %>% paste(collapse="+")
form <- formula(paste("nb_shares~",var.sel1,sep=""))
str_form <- paste("nb_shares~",var.sel1,sep="")
str_form
}
plot(reg.fit.forward,scale="bic",main="Forward BIC")
```

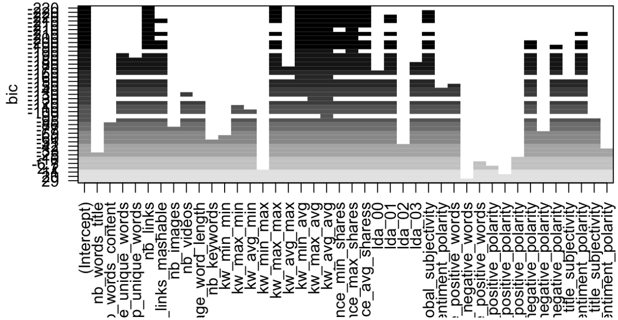
Forward BIC



Hide

```
plot(reg.fit.backward,scale="bic",main="Backward BIC ")
```

Backward BIC



Hide

```
subset.models <- data.frame(Criteria=rep(c("Bic"),each=2),Method=c("Forward","Backward"),Formula = c(g
et.formula(sum.reg.fit.f,nb.bic.f),get.formula(sum.reg.fit.b,nb.bic.b)))
subset.models
```

Criteria	Method
<fctr>	<fctr>

Criteria	Method
<fctr>	<fctr>
Bic	Forward
Bic	Backward

2 rows | 1-2 of 3 columns

We see that the variables selected with the forward or backward approaches are approximately the same.
For the forward approch, the BIC criteria selects the model:

$$Y = \beta_0 + nb.links * \beta_{nb.links} + kw.max.max * \beta_{kw.max.max} + kw.min.avg * \beta_{kw.min.avg}$$
$$+kw.max.avg * \beta_{kw.max.avg} + kw.avg.avg * \beta_{kw.avg.avg} + self.reference.max.shares * \beta_{self.reference.max.shares} + lda.03 * \beta_{lda.03}$$

We evaluate those models based on the RMSE

Hide

```
RMSE.list <- c()
RMSE.list <- append(RMSE.list, mean((pred.naive-data.test$nb_shares)^2)^(1/2))
RMSE.list <- append(RMSE.list, mean((predict(model.lm, newdata=data.test)-data.test$nb_shares)^2)^(1/2
))
for (formula in subset.models$Formula) {
  model.sub <- lm(formula,data=data.train)
  pred <- predict(model.sub, newdata = data.test)
  RMSE <- mean((pred-data.test$nb_shares)^2)^(1/2)
  RMSE.list <- append(RMSE.list,RMSE)
  }
RMSE.comparison <- data.frame(Criteria="mean",Method="Naive",Formula="Y")
RMSE.comparison <- rbind(RMSE.comparison,data.frame(Criteria="LM",Method="LM",Formula="Y"),subset.mode
ls)
RMSE.comparison <- RMSE.comparison %>% select(Criteria,Method) %>% mutate(RMSE=RMSE.list)
RMSE.comparison
```

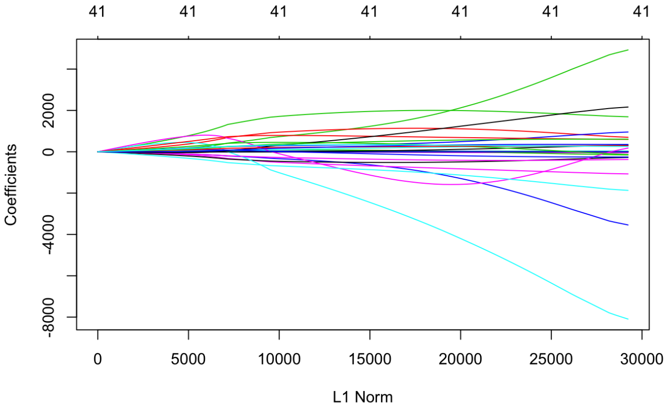
Criteria	Method	RMSE
<fctr>	<fctr>	<dbl>
mean	Naive	5472.156
LM	LM	5378.744
Bic	Forward	5364.830
Bic	Backward	5372.931

4 rows

Penalized regression

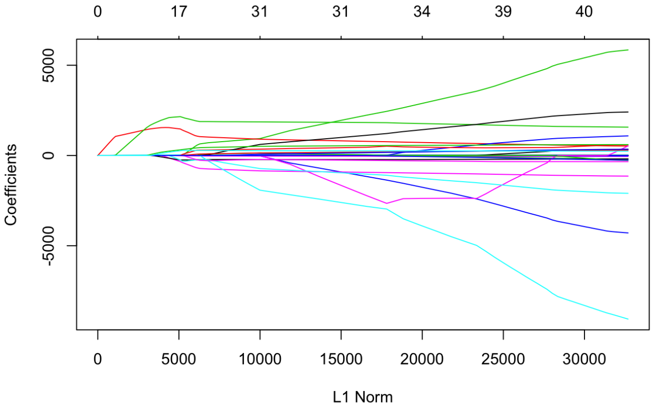
Hide

```
data.glmnet <- model.matrix(nb_shares~.,data=data.train)
model.ridge <- glmnet(data.glmnet,data.train$nb_shares,alpha=0)
model.lasso <- glmnet(data.glmnet,data.train$nb_shares,alpha=1)
plot(model.ridge)
```



Hide

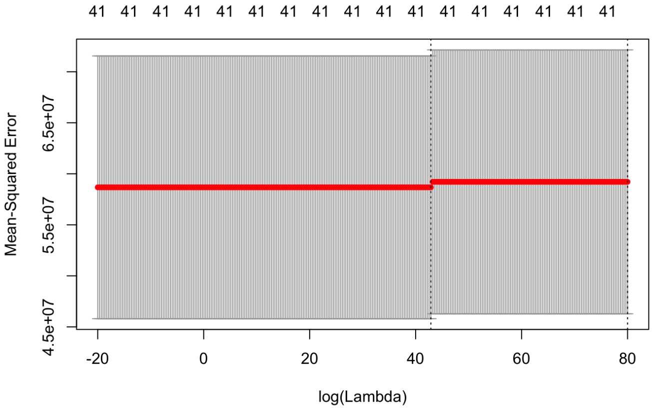
```
plot(model.lasso)
```



We can observe the regularization path for Lasso and Ridge Regression. For Lasso, some $\beta = 0$ for small t.

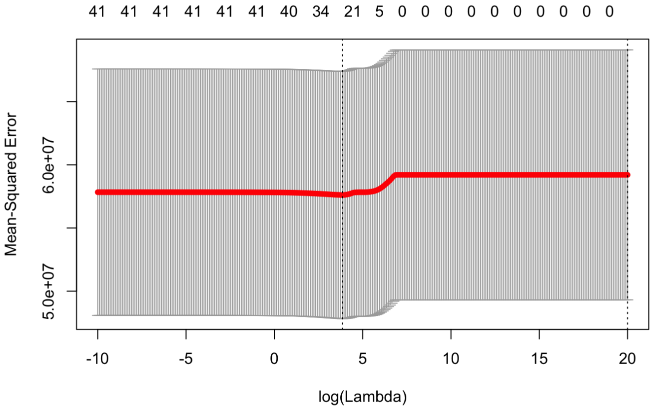
Hide

```
ridgeCV <- cv.glmnet(data.glmnet,data.train$nb_shares,lambda=exp(seq(-20,80,length=300)),alpha=0)
plot(ridgeCV)
```



Hide

```
lassoCV <- cv.glmnet(data.glmnet,data.train$nb_shares,lambda=exp(seq(-10,20,length=300)),alpha=1)
plot(lassoCV)
```



We look at the 2 best models

Hide

```
model.ridge <- glmnet(data.glmnet,data.train$nb_shares,lambda=ridgeCV$lambda.min,alpha=0)
model.lasso <- glmnet(data.glmnet,data.train$nb_shares,lambda=lassoCV$lambda.min,alpha=1)
RMSE.ridge <- mean((predict(model.ridge, newx=model.matrix(nb_shares~.,data=data.test))-data.test$nb_s
hares)^2)^(1/2)
RMSE.lasso <- mean((predict(model.lasso, newx=model.matrix(nb_shares~.,data=data.test))-data.test$nb_s
hares)^2)^(1/2)
RMSE.penalized <- data.frame(Criteria=paste("Lambda = ",ridgeCV$lambda.min),Method="Ridge",RMSE=RMSE.r
idge)
RMSE.penalized <- rbind(RMSE.penalized, data.frame(Criteria=paste("Lambda = ",lassoCV$lambda.min),Meth
od="Lasso",RMSE=RMSE.lasso))
RMSE.comparison <- rbind(RMSE.comparison, RMSE.penalized)
RMSE.comparison
```

Criteria <fctr>	Method <fctr>	RMSE <dbl>
mean	Naive	5472.156
LM	LM	5378.744
Bic	Forward	5364.830
Bic	Backward	5372.931
Lambda = 4177539791803456000	Ridge	5472.156
Lambda = 46.8126678173277	Lasso	5357.719

6 rows

Random Forests

We reduce the number and size of tree to make it compute in a timely manner

Hide

```
model.rf <- randomForest(nb_shares~.,data=data.train,nodesize=30,ntree=300)
model.rf
```

```

Call:
 randomForest(formula = nb_shares ~ ., data = data.train, nodesize = 30,      ntree = 300) 
               Type of random forest: regression
                     Number of trees: 300
No. of variables tried at each split: 13

          Mean of squared residuals: 59282165
                    % Var explained: -0.17
```

Hide

```
RMSE.rf <- mean((predict(model.rf, newdata=data.test)-data.test$nb_shares)^2)^(1/2)
RMSE.comparison <- rbind(RMSE.comparison, data.frame(Criteria="Node size = 30, ntree = 300",Method="R
F",RMSE=RMSE.rf))
RMSE.comparison
```

Criteria <fctr>	Method <fctr>	RMSE <dbl>
mean	Naive	5472.156
LM	LM	5378.744
Bic	Forward	5364.830
Bic	Backward	5372.931
Lambda = 4177539791803456000	Ridge	5472.156
Lambda = 46.8126678173277	Lasso	5357.719

Criteria <fctr>	Method <fctr>	RMSE <dbl>
Node size = 30, ntree = 300	RF	5561.902

7 rows

We increase parameters to get a more accurate prediction

Hide

```
model.rf2 <- randomForest(nb_shares~.,data=data.train,nodesize=1,ntree=400)
RMSE.rf <- mean((predict(model.rf2, newdata=data.test)-data.test$nb_shares)^2)^(1/2)
RMSE.comparison <- rbind(RMSE.comparison, data.frame(Criteria="Node size = 1, ntree = 400",Method="RF"
,RMSE=RMSE.rf))
RMSE.comparison
```

Criteria <fctr>	Method <fctr>	RMSE <dbl>
mean	Naive	5472.156
LM	LM	5378.744
Bic	Forward	5364.830
Bic	Backward	5372.931
Lambda = 4177539791803456000	Ridge	5472.156
Lambda = 46.8126678173277	Lasso	5357.719
Node size = 30, ntree = 300	RF	5561.902
Node size = 1, ntree = 400	RF	5607.964

8 rows

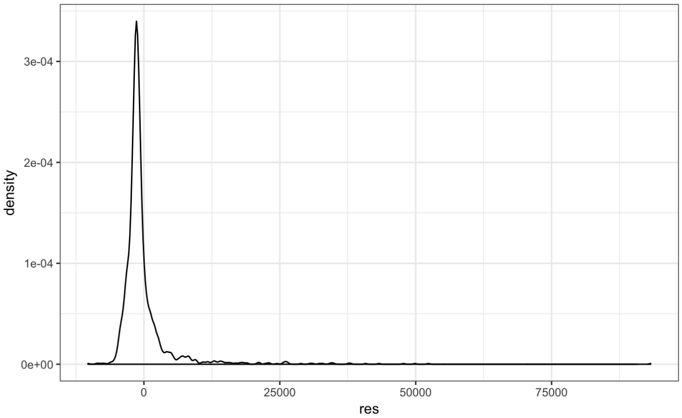
It doesn't improve the prediction, quite the contrary

7. Performances of each models

Based on our table of comparison of RMSE for each models, Lasso is the best performing model

Hide

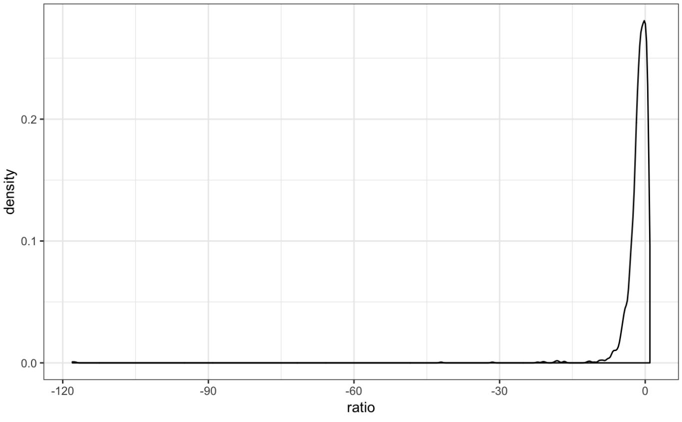
```
pred <- predict(model.lasso, newx=model.matrix(nb_shares~.,data=data.test))
obs <- data.test$nb_shares
res <- obs-pred
Prediction <- data.frame(pred, obs,res)
colnames(Prediction) <- c("pred","obs","res")
Prediction %>% ggplot() + aes(x=res) + geom_density()
```



Our model is quite optimistic (peak before 0 in residual's distribution), but still misses the few viral articles that get shared a lot. These are the ones that have a major impact on the poor RMSE score

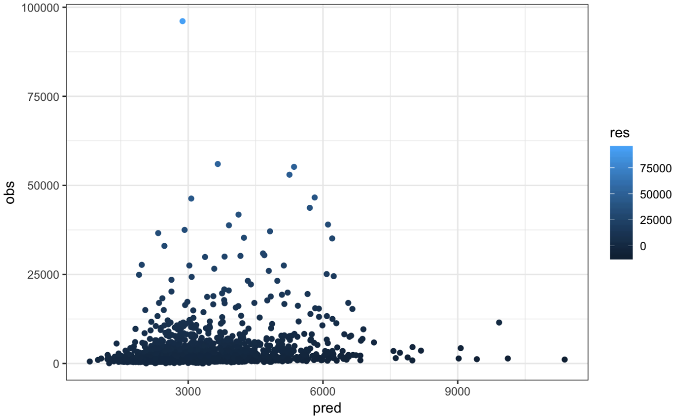
Hide

```
Prediction <- Prediction %>% mutate(ratio = res/obs)
Prediction %>% ggplot() + aes(x=ratio) + geom_density()
```



Hide

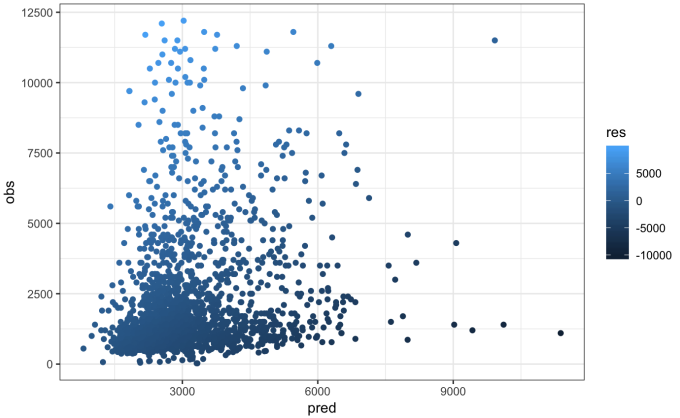
```
Prediction %>% ggplot() + aes(x=pred, y=obs, color=res) + geom_point()
```



We can see very few articles reach more than 12500 shares. They get very poor predictions as the model never predicts more than 11000 shares, and have a significant impact on the overall RMSE

Hide

```
Pred.trim <- Prediction[Prediction$obs < 12500,]
Pred.trim %>% ggplot() + aes(x=pred, y=obs, color=res) + geom_point()
```



Hide

```
mean(Pred.trim$res^2)^(1/2)
```

```
[1] 2372.71
```

If we had only observations with less than 12500 shares in the test data, the RMSE could have had been halfed. If on top of that we would not have viral articles in the train data, the RMSE would have been improved. However, because our goal was to predict the number of shares for all articles (including the ones with more than 12500 shares), we did not exclude popular articles from the database.

8. Conclusion

The popularity of an article depends on many characteristics: day of publication, length, channel, content… Using these characteristics as predictors, we have tried to predict the number of shares on social networks (popularity). Throughout analysis and modelling, we realise that there isn't a stable and standard recipe that will determine with a strong accuracy the popularity of an article. Our models can give an idea of the success but as Mashable has published articles making major buzz (more than 20 000 shares each; maximum reached with 840k shares), this twists our models and makes them more optimistic that they should be.

Furthermore, we could have transformed the regression problem in a classification problem, for example $nb.shares > 12000$**, that could have allow to create more robust model to predict whether an article would turn viral (meaning number of shares superior to 12000) or not. This model could potentially be combined afterwards with two different models dedicated to predicting number of shares for viral and for non-viral articles exclusively.**

To finish, we can keep in mind that the classification model is a lead to empower authors in defining what factors could be more impactfull to result in more engagement and virality.