Investigating the Popularity of Scientific Phenomena on Social Media Platforms

Sal Fu, Brian Lorenz, Jerry Xuan

Section 1: Motivation

Large numbers of scientific studies are published each day, many of which address pertinent gaps in human knowledge. Some results, however, become much more visible to the general populace than others. Why do some scientific results go viral, while others remain obscure? We wish to investigate what factors might influence the popularity of links about science on a social media platform, and the insights that may lend to scientific communication.

We choose Reddit as our platform for analysis because it is the 8th most visited site around the world, with over 500 million users total. This particular platform is convenient because it has a specific forum, the "Science" subreddit, that is dedicated to sharing and discussing scientific results. This particular subreddit can also reach a broad viewership because all Reddit users are subscribed to it by default (at the time this was written, the "Science" subreddit has around 18 million subscribers).

On Reddit, users can post content and "Upvote" the posts that they like. Within a subreddit, posts with higher upvotes will end up on the first page of the subreddit (e.g. they are the first posts that users will see when they visit the subreddit). When a post in "Science" gathers enough votes, it will end up on the "Reddit" frontpage, where it will get an even large viewership (i.e., all Reddit users will see the post when they open the website).

For this work, we will use "Upvotes" as our metric of whether a scientific article is popular: i.e., our response variable. We plan to build a random forest model to understand which variables may contribute the most to whether a scientific article garners a large number of votes.

The structure of this document is as follows: In section 2, we describe how we acquire data from Reddit, and how we use that data to acquire more information about the posts. In section 3, we describe the process of manipulating various columns to make our model more robust. In section 4, we describe the process of model building and interpretation. In Section 5, we summarize our results and interpret them in the context of our broader question.

Section 2: Data Acquisition and Modification

Using the Python Reddit API Wrapper, we obtain data about individual posts in the "Science" subreddit for the past two years (from Nov 2015 to Nov 2017). We query the number of upvotes that a post has, as that will be our metric of popularity. We also query for fields that represent explanatory variables, variables which could potentially be used to predict a post's popularity. For instance, these include the title of the post, the time at which the post was created, the amount of "karma" the post's author has, the number of comments on the post, the scientific subfield of the post, and so on.

With the fields that we queried as a basis, we extract additional information from them. For example, as we acquired data from the API, we create an explanatory variable called the "journal h index", which tracks posts that link to "high-impact" scientific articles. For this, we reference the h index rankings from Scimago Country and Rank (scimagojr.com), and define posts with a high h index as those that link to journals which are in the top 10 of the h index rankings. When attempting to download the data, some explanatory variables were not in a readable format for a .csv file, so they required encoding. However, a few characters in the past two years of data broke that encoding. As a result, we acquired the data from November 2015-November 2017 in 9 chunks. The results from the API query are saved in the repository as `reddit_df_final[number].csv`.

After we have generated this dataset, we work to extract additional information out of the post titles. First, we calculated the total length, mean word length, and max word length of each title. Then, we perform sentiment analysis on the titles to obtain a sentiment score for each title. All these became new explanatory variables for our data.

The title sentiment analysis works as follows: we obtain the corpus of words from SentiWordNet, which assigns nearly every word in the English language a positive or negative index with a value between 0 and 1. From each of our post titles, we remove "stopwords" like "about" and "the" (words which do not have positive or negative connotations). Taking words with an overall positive meaning to have a positive value, and treating negatively-connotating words analogously, we sum up the total indices of the remaining words in the title. We then divide that sum by the number of words in the title to acquire our desired variable, which can be thought of as the title's sentiment "density".

We save the results of this analysis in the repository as `reddit_df_final[number]_sent.csv`. In the code below, we read in the fields:

```r
#Reading in the packages that we used to wrangle our data

require(lubridate)
require(dplyr)
library(readr)
require(ggplot2)
library(stringr)
```

```r
#Read the reddit data into a dataframe
reddit_df_final1 <- read_csv("reddit_df_final1_sent.csv")
reddit_df_final2 <- read_csv("reddit_df_final2_sent.csv")
reddit_df_final3 <- read_csv("reddit_df_final3_sent.csv")
reddit_df_final4 <- read_csv("reddit_df_final4_sent.csv")
reddit_df_final5 <- read_csv("reddit_df_final5_sent.csv")
reddit_df_final6 <- read_csv("reddit_df_final6_sent.csv")
reddit_df_final7 <- read_csv("reddit_df_final7_sent.csv")
reddit_df_final8 <- read_csv("reddit_df_final8_sent.csv")
reddit_df_final9 <- read_csv("reddit_df_final9_sent.csv")

reddit_df <- rbind(reddit_df_final1,reddit_df_final2,reddit_df_final3,
                   reddit_df_final4,reddit_df_final5,reddit_df_final6,
                   reddit_df_final7,reddit_df_final8,reddit_df_final9)

#Store a copy of the dataframe for later use
reddit_df_read <- reddit_df
```

We wanted to get a sense for whether sentiment analysis worked as expected, so we looked at the titles with the most "positive" sentiment and the most "negative" sentiments; the code below prints out these values:

```r
reddit_df_read %>%
  arrange(desc(title_sent)) %>%
  select(title_sent,title) %>%
  head()
```

```
## # A tibble: 6 x 2
##   title_sent
##        <dbl>
## 1      0.625
## 2      0.625
## 3      0.500
## 4      0.500
## 5      0.475
## 6      0.450
## # ... with 1 more variables: title <chr>
```

The most positive two titles are: "Nonnutritive sweeteners and cardiometabolic health" and "Babies know when they don't know something". Some of these are reasonable? But others are: "Happiness doesn't bring good health, study finds," and "Facebook posts inspired by envy, UBC study finds." For some of these, we can see where the sentiment analysis failed. For instance, "Happiness doesn't bring good health" has positive words like "happiness," "good," and "health", which can make the average sentiment of the title positive even if the presence of the single negation "doesn't" alters the overall meaning of the sentiment.

Printing the posts with the most negative title sentiments:

```
reddit_df_read %>%
  arrange(title_sent) %>%
  select(title_sent,title) %>%
  head()
```

```
## # A tibble: 6 x 2
##   title_sent
##        <dbl>
## 1 -0.6250000
## 2 -0.5625000
## 3 -0.5000000
## 4 -0.5000000
## 5 -0.5000000
## 6 -0.4583333
## # ... with 1 more variables: title <chr>
```

Sentiment analysis was a great exercise, but latter when we computed variable importance for our random forest model, we see that its lack of robustness meant that it was not a meaningful predictor for our response variable.

Section 3: Variable Modification

After retrieving the data, we take multiple steps to wrangle it into a format that makes more sense given the variables we are trying to investigate. The data for "Time" that we retrieve from is in units of seconds elapsed from January 1st, 1970 in UTC time. We thus use the code below to translate that metric of time into year, date, and hour.

```
#Store all of the POSIX dates into a vector
dates <- as.POSIXct(reddit_df$created_utc,origin = "1970-01-01",tz = "UTC")

#Use lubridate to read y,m,d,h from that vector. Create new columns for these
reddit_df <- reddit_df %>%
  mutate(post_year = year(dates),post_month = month(dates), post_day = day(dates),
         post_hour = hour(dates))
```

After obtaining that information, we create a categorical variable for "time" to categorize the time of day that the post was created into "Day", "Morning", and "Night". This step was necessary because otherwise, the classification algorithm would not classify "23" and "0" as the same portion of the day (time wise), which would change with our results. Furthermore, having 24 categories for hour would not be ideal for CART, which random forests are built on.

```
#Create a new column for categorical time of day
#Time ranges: PST 6pm-3am night      3am-10am morning    11am-6pm day
mk_cat_time <- function(h)
```

```
{
  ifelse(h>18, "night", ifelse(h>10, "day", ifelse(h>2, "morning", "night")))
}

#Create a new column for categorical time of month
#Time ranges: 1-10 early    11-20 mid    21-31 late
mk_cat_day <- function(d)
{
  ifelse(d>20, "late", ifelse(d>10, "mid", "early"))
}

#Create a new column for categorical time of year
#Time ranges: Dec-Feb: Winter   Mar-May: Spring    Jun-Aug: Summer    Sep-Nov: Fall
mk_cat_month <- function(m)
{
  ifelse(m>11, "Winter", ifelse(m>8, "Fall",
                                ifelse(m>5, "Summer", ifelse(m>2, "Spring", "Winter"))))
}

reddit_df <- reddit_df %>%
  mutate(cat_post_hour = mk_cat_time(post_hour)) %>%
  mutate(cat_post_day = mk_cat_day(post_day)) %>%
  mutate(cat_post_month = mk_cat_month(post_month))
```

Next, we clean up the data in order to remove posts that do not share scientific content (e.g. AMAs, subreddit discussions) by filtering under the subfield column. We also remove all subfields that have less than 100 posts, since these were mostly errorneously named subfields that had only 1 post.

```
#Filter by removing the Ask Me Anything (AMA) threads and the subreddit discussion
  #threads since these are not studies.
reddit_df <- reddit_df %>%
  filter(!str_detect(subfield, " AMA")) %>%
  group_by(subfield) %>%
  filter(n() > 100)

#Display the remaining subfields
count(reddit_df,subfield)
```

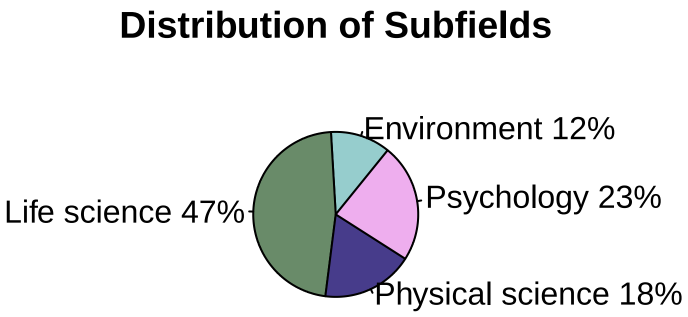
```
## # A tibble: 20 x 2
## # Groups:   subfield [20]
##            subfield     n
##               <chr> <int>
##  1   Animal Science  1214
##  2     Anthropology   544
##  3        Astronomy   811
##  4          Biology  2824
##  5           Cancer   606
##  6        Chemistry   483
##  7 Computer Science   234
##  8    Earth Science   542
##  9      Engineering   541
## 10      Environment  1949
## 11     Epidemiology   364
## 12          Geology   495
## 13           Health  2434
## 14         Medicine  2001
## 15      Nanoscience   326
## 16     Neuroscience  1716
## 17     Paleontology   531
## 18          Physics   917
## 19       Psychology  1694
## 20   Social Science   963
```

Now, we group the remaining subfields into four major categories so that our model can more easily divide among them. We made these selections by our impressions of the categories that these fields fall into, so this step is certainly subjective.

```
#Make a function that sorts fields into "Life Science,"
#"Psychology," "Physical Science," "Environment"
mk_cat_subfield <- function(field)
{
  ifelse(field == "Biology" | field =="Health" | field =="Cancer" |
           field =="Medicine" | field =="Animal Science" | field =="Paleontology" |
           field =="Epidemiology", "Life Science", ifelse(field =="Psychology" |
           field =="Neuroscience" | field =="Social Science" |
            field =="Anthropology", "Psychology",
           ifelse(field =="Geology" | field =="Astronomy" | field =="Physics" |
            field =="Chemistry" | field =="Nanoscience" | field =="Engineering" |
            field =="Computer Science", "Physical Science", "Environment")))
}
#Apply the function and remove the old column
reddit_df <- reddit_df %>%
  mutate(cat_subfield = mk_cat_subfield(subfield)) %>%
  select(-subfield)

#Turn the new column into a factor for model use
reddit_df$cat_subfield <- factor(reddit_df$cat_subfield)
```

Here we make a pie plot of the distribution of articles into these four categories:

```
subfield_freq = table(reddit_df$cat_subfield)
type_freq_perc = paste(round((subfield_freq/length(reddit_df$cat_subfield)),2)*100,
                       "%",sep="")
type_freq_lab = paste(c("Environment","Life science",
                        "Physical science", "Psychology"), type_freq_perc, sep = " ")
#png("subfield_chart.png", width=8, height=6, units="in", res=500)
pie(subfield_freq,labels=type_freq_lab, init.angle = 51,edges = 10000,
    col = c("paleturquoise3","darkseagreen4","slateblue4","plum2"), main="Distribution of Subfields")
```



```
#dev.off()
```

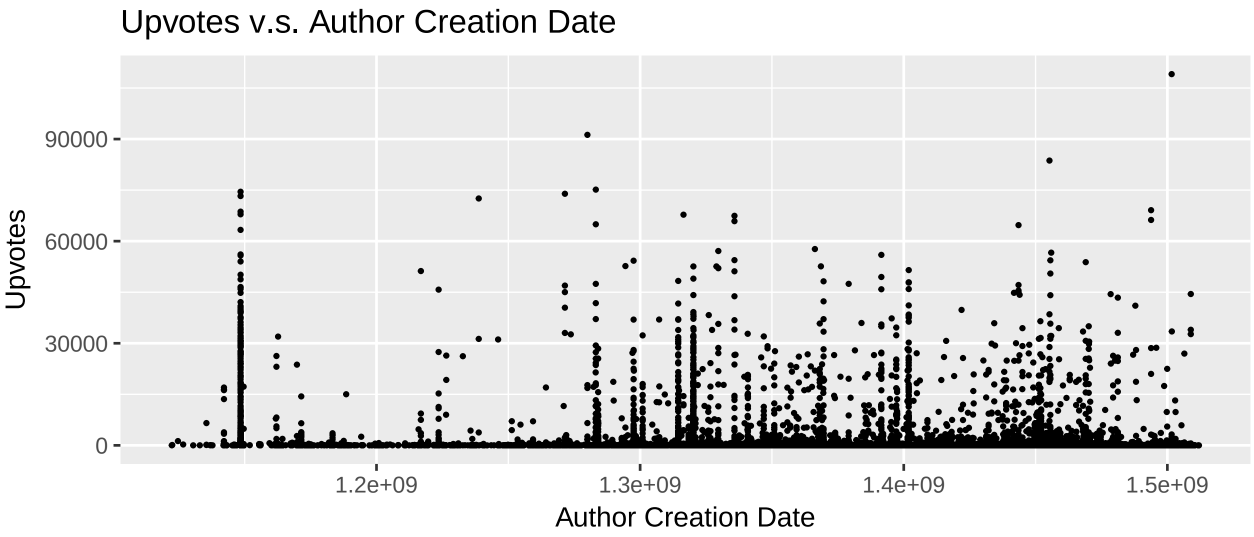
Here we turn the categorical variables to factors, which the caret random forest model expects.

```
#Turn all categorical variables into factors
reddit_df$author_flair_binary <- factor(reddit_df$author_flair_binary)
reddit_df$image <- factor(reddit_df$image)
reddit_df$journal_h_index <- factor(reddit_df$journal_h_index)
reddit_df$post_year <- factor(reddit_df$post_year)
reddit_df$cat_post_month <- factor(reddit_df$cat_post_month)
reddit_df$cat_post_hour <- factor(reddit_df$cat_post_hour)
reddit_df$cat_post_day <- factor(reddit_df$cat_post_day)
```

Next we plot select explanatory variables against our response variable to get a sense of the correlations.

Author created date vs upvotes. We see a few vertical lines of dots (e.g. the one near 1.15e+09), which we looked into and realized that they are single authors that are prolific on Reddit! Since these outliers could bias our model's prediction vis-a-vis author_created_date, we later remove that column from the dataframe.

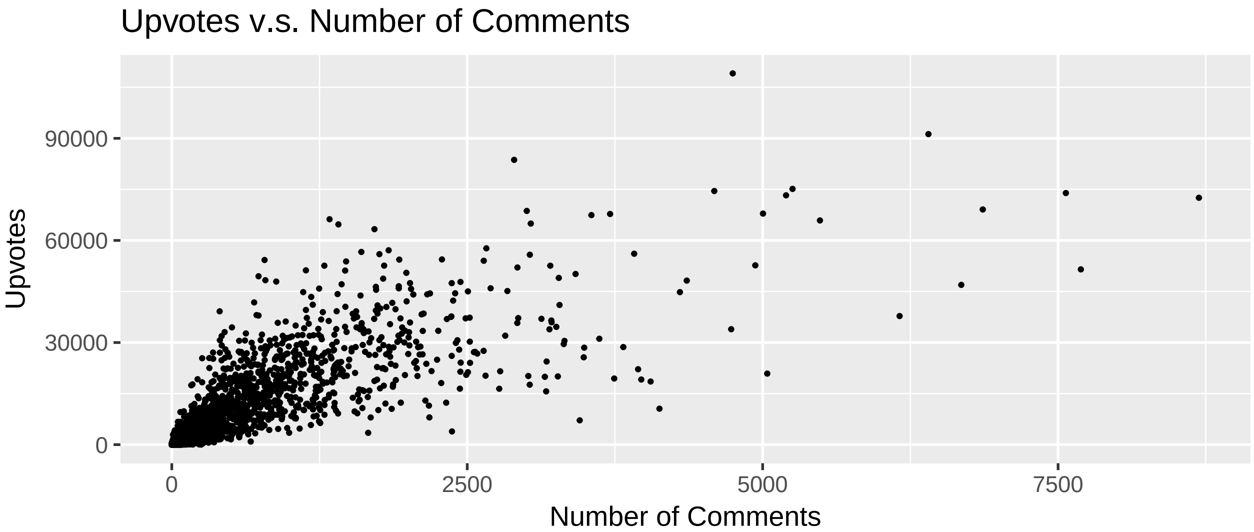
```
acd_plot <- ggplot(data=reddit_df, aes(x=author_created_date, y=upvotes))+
  geom_point(size=0.5) +
  xlab("Author Creation Date")+
  ylab("Upvotes")+
  ggtitle("Upvotes v.s. Author Creation Date")
#ggsave(acd_plot, filename='./acd_plot.png', width=8, height=5)

acd_plot
```



Number of comments vs upvotes. This is an example of a variable that shows strong positive correlation with our response variable. Indeed, we could have used number of comments instead of upvotes as our response variable and achieved a very similar model. To create a fair and meaningful prediction model, we thus remove num_comments and other highly correlated variables (link_karma,comment_karma, and gilded).

```
nc_plot <- ggplot(data=reddit_df, aes(x=num_comments, y=upvotes))+
  geom_point(size=0.5) +
  xlab("Number of Comments")+
  ylab("Upvotes")+
  ggtitle("Upvotes v.s. Number of Comments")
```

```
#ggsave(nc_plot, filename='./nc_plot.png', width=8, height=5)

nc_plot
```



```
#To prepare for model building, we remove unnecessary columns and ones correlated with response
reddit_df <- reddit_df %>%
  select(-X1,-author,-id,-created_utc,-domain,-url,-author_flair,-post_day,
         -post_hour,-post_month,-num_comments,-gilded,-link_karma,
         -comment_karma,-title,-X1_1,-author_created_date)
```

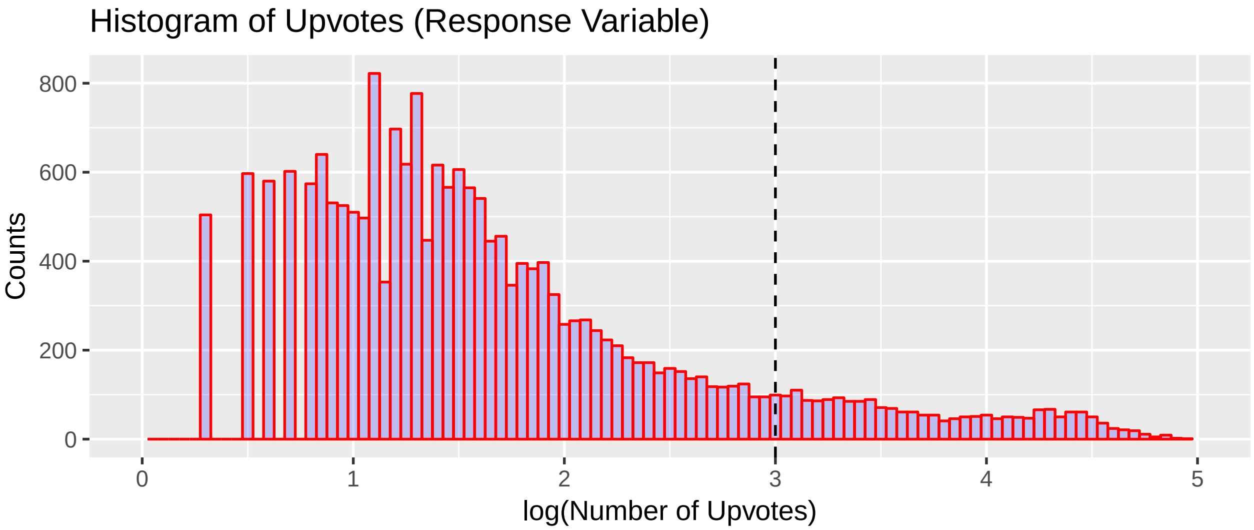
Now we remove any rows that have missing values (NAs). We also print the proportion of rows removed below to ensure that we are not removing a large fraction of the data. It is comforting to see that only 0.02% of our observations contained any NAs.

```
bef_rm <- nrow(reddit_df)

#Remove missing values
reddit_df <- reddit_df[complete.cases(reddit_df),]
aft_rm <- nrow(reddit_df)

paste("Proportion of rows removed: ",(bef_rm-aft_rm)/bef_rm)
```

```
## [1] "Proportion of rows removed:  0.000235971494643447"
```

Here we plot a histogram of the number of upvotes, showing that the vast majority of our data has "low" (<1000) upvotes. The huge range (from 0 to 100000), and high level of skewedness of upvotes motivate us to separate our response into a binary variable.

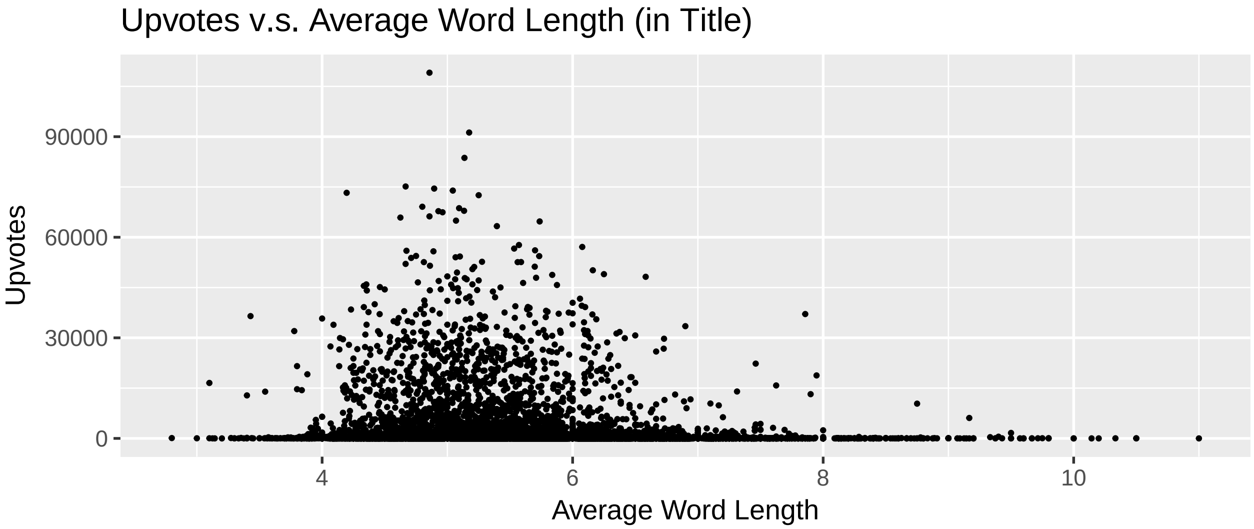
```
hist_upvotes <- qplot(log10(reddit_df$upvotes+1),
      geom="histogram",
      binwidth = 0.05,
      main = 'Histogram of Upvotes (Response Variable)',
      xlab = "log(Number of Upvotes)",
      ylab = "Counts",
      fill=I("blue"),
      col=I("red"),
      alpha=I(.2),
       xlim=c(0,5)) +
  geom_vline(xintercept=3,show_guide=TRUE,linetype="dashed")

hist_upvotes
```



```
#ggsave(hist_upvotes, filename='./hist_upvotes.png', width=8, height=5)
```

Here, we plot the mean title word length vs. number of upvotes. There appears to be a sweet spot around 5. It appears that lower mean title word lengths give higher upvotes. There are, however, more posts near the low range too, so we cannot make that conclusion just yet.

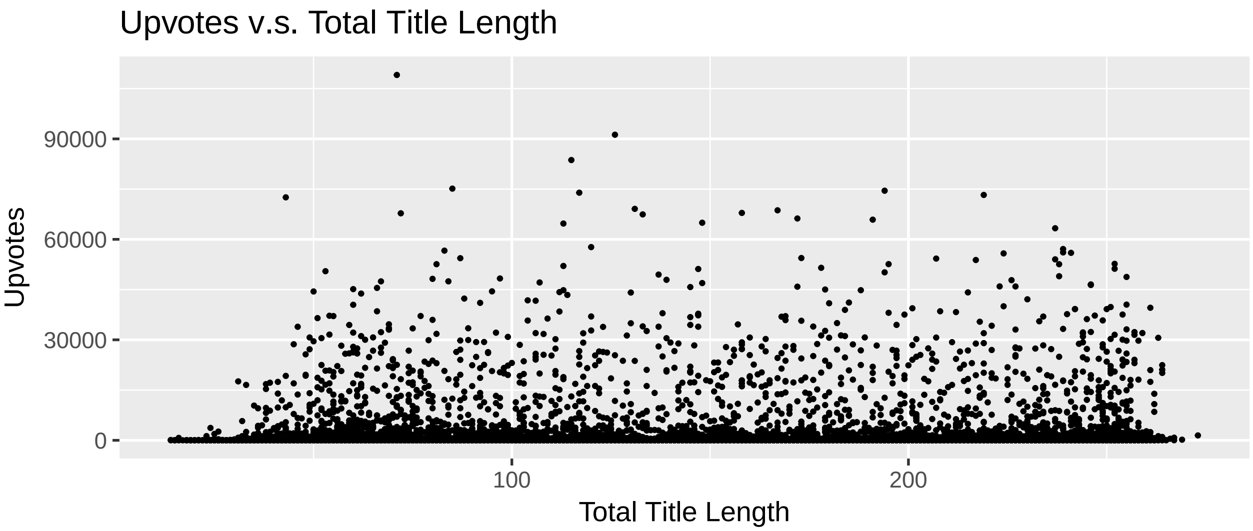
```
mtl_plot <- ggplot(data=reddit_df, aes(x=mean_title_length, y=upvotes))+
  geom_point(size=0.5) +
  xlab("Average Word Length")+
  ylab("Upvotes")+
  ggtitle("Upvotes v.s. Average Word Length (in Title)")

mtl_plot
```



```
#ggsave(mtl_plot, filename='./mtl_plot.png', width=8, height=5)
```

Here, we plot the total title length vs. number of upvotes. No strong correlation could be seen.

```
tl_plot <- ggplot(data=reddit_df, aes(x=title_len, y=upvotes))+
  geom_point(size=0.5) +
  xlab("Total Title Length")+
  ylab("Upvotes")+
  ggtitle("Upvotes v.s. Total Title Length")
ggsave(tl_plot, filename='./tl_plot.png', width=8, height=5)

tl_plot
```



Now we turn the continuous upvote response variable into a binary response. We choose 1000 upvotes as our cut-off, because that is the approximate threshold where a post will end up on the front page of the Science subreddit, from which it could reach the front page of the entire website.

```
#Create a new column for categorical upvotes
#Ranges: 1000+ high      0+ low
mk_cat_upvotes <- function(upv)
{
  ifelse(upv>1000, "High", "Low")
}
#Apply the function and remove the old column
reddit_df <- reddit_df %>%
  mutate(cat_upvotes = mk_cat_upvotes(upvotes)) %>%
  select(-upvotes)

#Turn the column into a factor
reddit_df$cat_upvotes <- factor(reddit_df$cat_upvotes)
```

Section 4: Model Building and Assessment

After wrangling the data, we are ready to build our Random Forest model. We partition the data into test and training sets, setting aside the test data so that we can appropriately assess our model's accuracy.

```
#Packages required for model building
require(caret)
require(rpart)

#Set the seed for reproducibility
set.seed(47)

#Split the data into test and training, with 80% going to training
inTrain <- createDataPartition(y = reddit_df$cat_upvotes, p=0.80, list=FALSE)
reddit.train <- reddit_df[inTrain,]
reddit.test <- reddit_df[-c(inTrain),]
```

By looking at the number of observations in each category below, we see that predicting everything as "low" gives a fairly high accuracy. In other words, for the model, there is little consequence in masclassifying the posts with "high" upvotes. To avoid this problem in building our model, we sample from our training set to have an equal number of observations with "high" and "low" upvotes.

```
#Print the number of observations in each category
sum(reddit.train['cat_upvotes'] == "High")
```

```
## [1] 1724
```

```
sum(reddit.train['cat_upvotes'] == "Low")
```

```
## [1] 15224
```

```
### Sample data such that there is a equal number of observations for each category.

set.seed(4747)

#Count the number of high upvotes
num_high_upvote <- sum(reddit.train['cat_upvotes'] == "High")

reddit.train.high <- subset(reddit.train, reddit.train['cat_upvotes'] == "High")
reddit.train.low <- subset(reddit.train, reddit.train['cat_upvotes'] == "Low")

#Select an equal proportion of high and low upvotes
inTrain.low <- createDataPartition(y = reddit.train.low$cat_upvotes,
                                   p=num_high_upvote/dim(reddit.train.low)[1],
                                   list=FALSE)
reddit.train.low <- reddit.train.low[inTrain.low,]
```

```
### Combine the 2 dataframes into the final training data
reddit.train.final <- rbind(reddit.train.low, reddit.train.high)
```

```
#Check that the numbers are equal.
reddit.train.final %>% group_by(cat_upvotes) %>% summarize(n())
```

```
## # A tibble: 2 x 2
##   cat_upvotes `n()`
##        <fctr> <int>
## 1        High  1724
## 2         Low  1724
```

Now we grow the random forest, using the out of bag (OOB) error rate to tune the `mtry` parameter (number of variables at each split).

```
#Build the random forest model
set.seed(4747)

rf.reddit <- train(cat_upvotes ~., data=reddit.train.final, method="rf",
                      trControl = trainControl(method="oob"),
                      ntree=1000, tuneGrid = data.frame(mtry=c(3,5,7,9,11)),
                      importance = TRUE,na.action = na.exclude)
```

```
rf.reddit
```

```
## Random Forest
##
## 3448 samples
##   13 predictor
##    2 classes: 'High', 'Low'
##
## No pre-processing
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    3    0.6502320  0.3004640
##    5    0.6432715  0.2865429
##    7    0.6400812  0.2801624
##    9    0.6429814  0.2859629
##   11    0.6377610  0.2755220
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 3.
```

```
rf.reddit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 1000, mtry = param$mtry, importance = TRUE)
##               Type of random forest: classification
##                     Number of trees: 1000
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 35.93%
## Confusion matrix:
##      High Low class.error
## High 1211 513   0.2975638
## Low   726 998   0.4211137
```

The model with the best accuracy corresponds to `mtry` = 3. Using this best model, we predict our test data to assess our model's accuracy.

```
#mean((log(reddit.test$upvotes+1) - predict(rf.reddit, newdata = reddit.test))^2)
confusionMatrix(data=predict(rf.reddit, newdata = reddit.test), reference = reddit.test$cat_upvotes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low
##       High  297 1530
##       Low   134 2275
```

```
##
##                Accuracy : 0.6072
##                  95% CI : (0.5923, 0.6219)
##     No Information Rate : 0.8983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1178
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.68910
##             Specificity : 0.59790
##          Pos Pred Value : 0.16256
##          Neg Pred Value : 0.94438
##              Prevalence : 0.10175
##          Detection Rate : 0.07011
##    Detection Prevalence : 0.43130
##       Balanced Accuracy : 0.64350
##
##        'Positive' Class : High
##
```

The confidence interval for the accuracy of the model is between 0.5923 and 0.6219, which is greater than 50%: this suggests that there *is* information, i.e., that there is a basis on which to distinguish opular posts from less popular posts. With that in mind, we print the variable importance from the model, which indicates which explanatory variables most directly impact the response:

```
varImp(rf.reddit,scale=FALSE)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 38)
##
##                              Importance
## title_len                        31.452
## imageyes                         21.267
## mean_title_length                19.746
## journal_h_indexlow               17.570
## subfieldGeology                  15.876
## cat_post_hournight                8.193
## subfieldHealth                    7.691
## author_flair_binaryyes            7.378
## subfieldPsychology               7.191
## subfieldBiology                   6.351
## max_title_length                  5.773
## subfieldNanoscience               5.111
## cat_post_monthSummer              4.530
## cat_subfieldPhysical Science      4.136
## subfieldCancer                    3.770
## subfieldNeuroscience              3.441
## cat_subfieldPsychology            3.429
## post_year2017                     2.724
## subfieldEngineering               2.629
## subfieldChemistry                 2.574
```

Of the variables that we use in our model, the output above suggests that the five most important variable contributing to the popularity of a post are: 1. Length of the title of the post, 2. Whether there is a thumbnail accompanying the post, 3. The average length of words in the title, 4. The h index of content linked to in the post, and 5. Whether the post was published between the hours of 6PM-3AM PST (nighttime).

The first one might be because post titles that summarize the content of the article are better. Thumbnails give an article more legitimacy, so the second one also makes sense. The third and the fourth should be interpreted together. Whether a journal has a low h index might correspond to whether it was a pop science article, which in turn might be shorter words in the title because –> this also makes sense, because easier for users to understand. It also seems important to post

The next two important variables after the first 5 are whether the post is in health, biology– which we might expect, since people might be more inclined to pay attention to studies that are more pertinent to their condition (health is a topic that easily links to elements of the human condition).

Section 5: Conclusions

In this project, we attempt to look for variables that may affect whether a science post on Reddit gathers a large number of upvotes. From the results of the previous section, we find that the most important variables are post title length, whether there is a thumbnail, the average length of a word in the title, whether the post links to a high-impact scientific journal, and whether the the post was published at nighttime PST. It seems that whether the post.

overall, these results suggest a kind of "fast-food" model of scientific media consumption, in which users want to get the gist of the post. The variables suggest that the posts that resonate with users are the ones that convey the point of the post quickly. Posts with subfields that are more relevant or relateable may also contribute to the popularity of a post.

Data generalizability: something about how Reddit isn't representative of the population, but the results generally make sense given our experiences with scientific communication. seems wishy washy though because the extent of our project doesn't let us go into that much more detail... segue into next small section

What held us back: not being able to get more information from our posts, especially from the specifics of post title or even post content (because we did have access to the URL) –> sentiment analysis fell short, tackling the other problem would've required more time than we probably had

Concluding remarks: something that links abck to the motivating question... wax eloquence here