Personalized Medicine:Redefining Cancer Treatment

Aim: Automatically classify genetic mutations that contribute to cancer tumor growth (so-called “drivers”) in the presence of mutations that don’t affect the tumors (“passengers”).

About: A cancer tumor can have thousands of genetic mutations. But the challenge is to classify the mutations that contribute to tumor growth (drivers) from the neutral mutations (passengers). Currently this interpretation of genetic mutations is being done manually. This is a very time-consuming task where a clinical pathologist has to manually review and classify every single genetic mutation based on evidence from text-based clinical literature. In this project we are going to build a machine learning algorithm that using this knowledge base as a baseline, automatically classifies genetic variations.

[Exploratory Analysis - Instacart](https://www.kaggle.com/philippsp/exploratory-analysis-instacart)

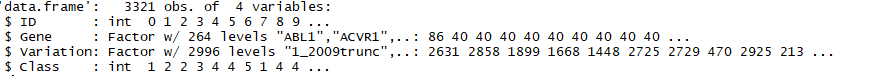
ABOUT DATA:

We have data in 4 different files , two csv files and two text files:

* training/test variants: These are the csv files of the gene mutations together with the target value Class, which is the (manually) classified assessment of the mutation.  The feature variables are Gene, the specific gene where the mutation took place, and Variation, the nature of the mutation. The test data doesn’t contain the class values, but to predict those classes. These two files each are linked through an ID variable .
* training/test text: These files contain the detail description of the evidence that was used by the experts to label the mutation classes.

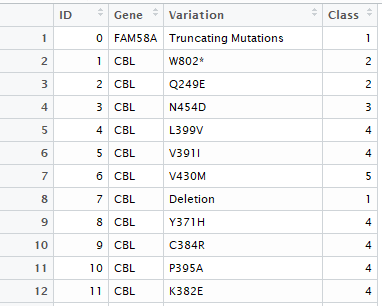
training variants.csv: This file contain 4 variables and above 3000 observations.

Detail of train variants:

Description of all the train variables:

* **ID**: The id of the row used to link the mutation to the clinical evidence
* **Gene**: The gene where the genetic mutation is located
* **Variation:** The aminoacid change for this mutations
* **Class:** 1-9 the class this genetic mutation has been classified on

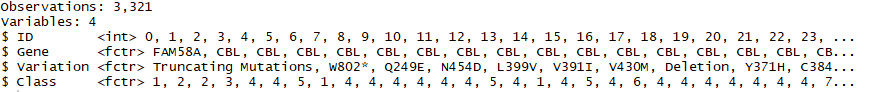
View of training variants**:**



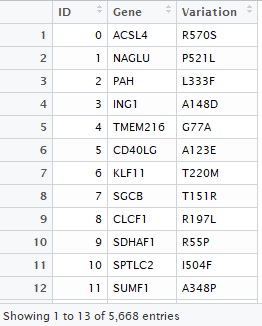
Need to convert the data type of class variable into factor,

train\_data$Class=as.factor(train\_data$Class)

glimpse(train\_data)



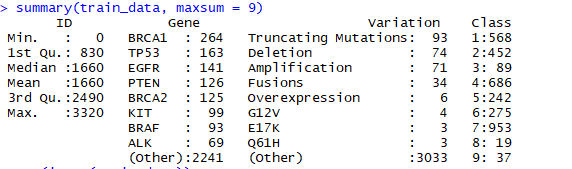
View of training variants:



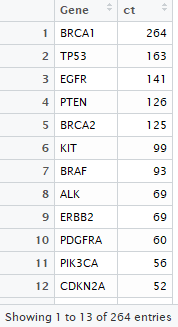
No n.a value present in training variants,

F:\shah\snapshots\train_variants_na.PNG

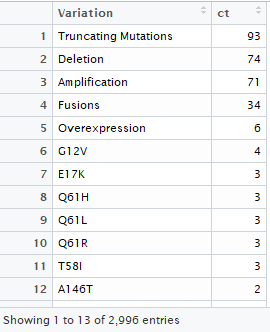
Summary detail of all the variables of training variants,



summarizing the training variants data by Gene variable,



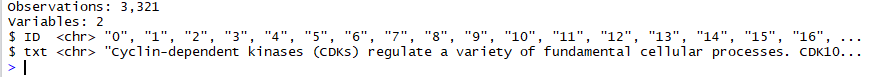
Summarizing the training variants by variation variable,



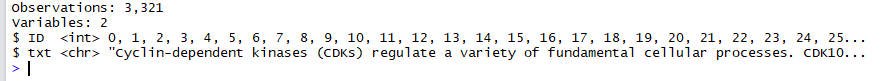
training\_text:  This text file contains the clinical evidence (text) used to classify genetic mutations and it is delimited with double pipe(||).The file contains the same number of observations as training variants but only 2 variables(ID and text(delimited by ||)).

Overview of training\_text:

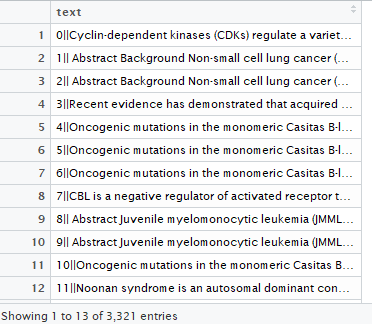
glimpse(train\_text)



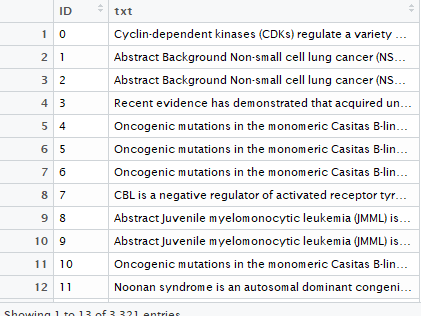
After converting ID variable into proper data type



Initially,



After separating ID and text variables,

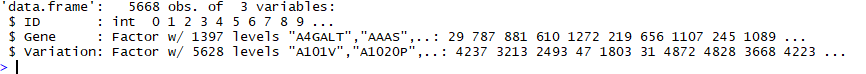


Description of all the training text variables:

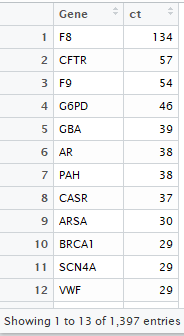
* **ID**: The id of the row used to link the clinical evidence to the genetic mutation
* **text**: The clinical evidence used to classify the genetic mutation

test\_variants.csv : This csv file containing the description of the genetic mutations used for training. It is having 3 variables and some what five and a half thousand observations. The absence of the class variable in the test is obvious, that we have to predict. The description of all the test variants is same as training variants that I explained above.

Detail of test variants:

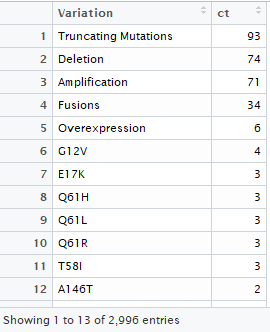
we can observe, the number of observations in the test variants are greater than those in training variants. The data description tells us that “Some of the test data is machine-generated to prevent hand labeling.”, which should explain this otherwise curious imbalance.

Summary of each variable of test variants by Gene variable,



The most frequent Genes in the training variants compared to test variants data are completely different. Moreover, the test data seems to contain significantly more different Genesand fewer high-frequency Genes than the train data. To some extent, this might be an effect of the added machine-generate entries in the test data

Summary of each variable of test variants by variation variable,

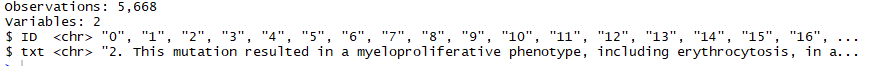


The most frequent Variationsin training variants compared to test variants are very much identical. Although, again, the corresponding frequencies are lower in the test data.

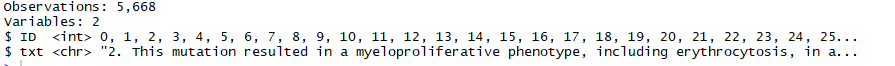
test\_text: This text file contains the clinical evidence(text) used to classify genetic mutations and it is delimited with double pipe(||).The file contains the same number of observations as training variants but only 2 variables(ID and text, delimited by ||).

Overview of test\_text:

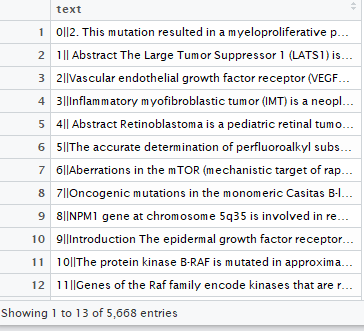
glimpse(test\_text)



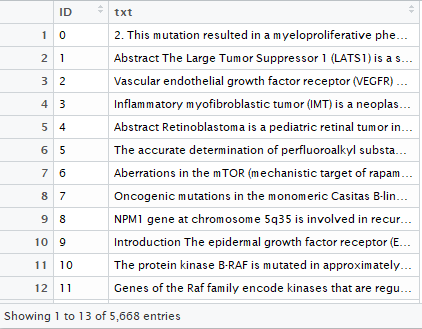
After converting ID variable into proper data type



Initially,



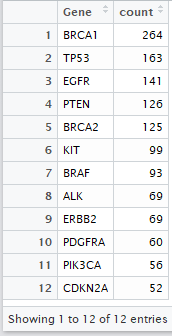
After separating ID and text columns,



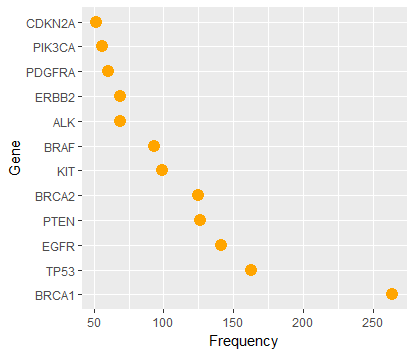
Feature visualization(For training/test variants)

For training variants,

Most frequent Gene,

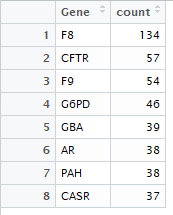


geom\_plot for most frequent Gene,

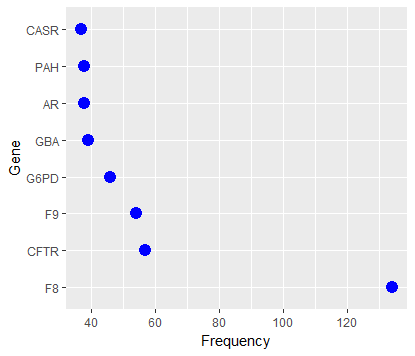


For test variants,

Most frequent Gene in test variants,

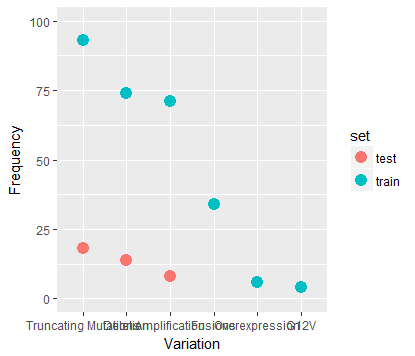


Geom plot,



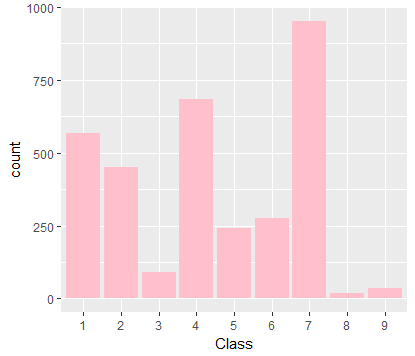
From the above two plots we can easily conclude that,most of the Gene levels having realatively low frequency for both train and test.And the test data has fewer high-frequency Genes.

Most frequent variation for train(blue) and test(red),



We can observe with the plot that the variation is same for both train and test but the frequencies are completely different (we already saw through tables as well).

Train **Class** target distribution,



Class 7 is clearly the most frequent one, while classes 1,2 and 4 are of moderate frequency. And the classes 3, 8 and 9 are the less frequency getter.

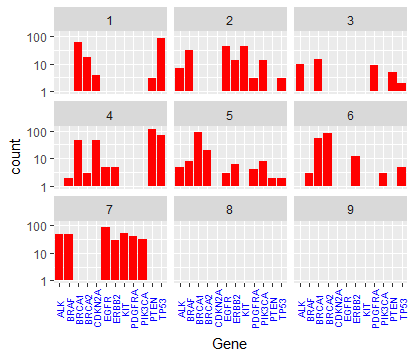
Feature communication

Comparison between Gene and class.

Here we want to examine how the features interact with each other and with the target class variable.

Firstly we are trying to know, how the most frequent Gene is varying with the different levels of class target.

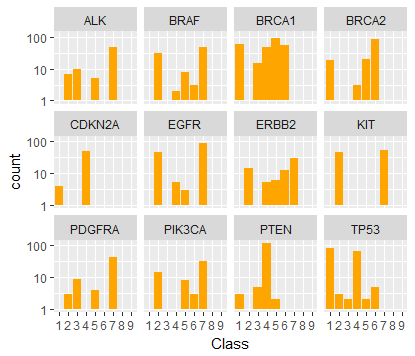
Plot on Logarithmic frequency scales,



“PTEN” Gene is predominantly present in a single Class 4,“TP53”, is mainly shared between 2 classes 1 and 4,while Classes 8 and 9 contain none of the most frequent Genes.

Now I want to know how all the class levels are varying with sorted Gene,

Again we are using logarithmic frequency scales,

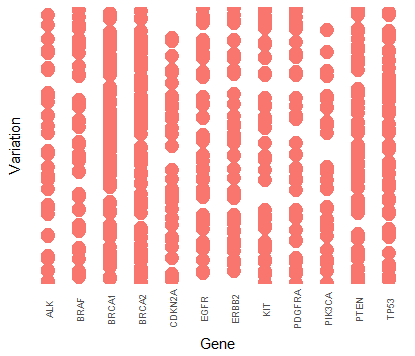


We can clearly see the observations that we made for the previous plot. That is, ”PTEN” is predominantly present in class 4,”TP53” is sharing between the classes 1 and 4.And the class 8 and 9 having no significance for any Gene.

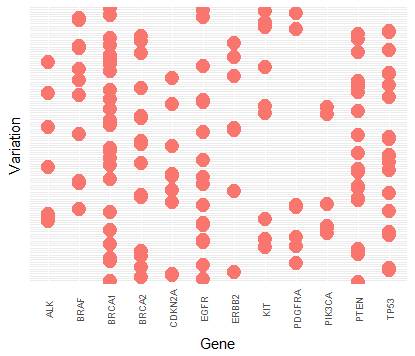
Comparison between Gene and variation

Here I try to observe how the variations are distributed for the most frequent Gene by using Geom\_count() function.

For training variants:



For test variants:



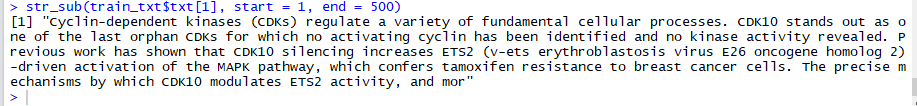
By analyzing both the above plot I find the two datasets (training/test variants)are diverse one again.

Text Files(train/test):

These data files contain lots of text from what looks like scientific papers or proceedings.

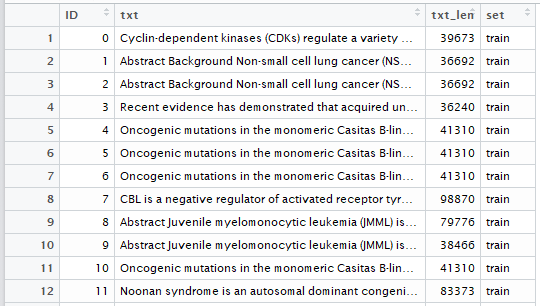
Few words of the first text of train\_txt are :

str\_sub(train\_txt$txt[1], start = 1, end = 500)



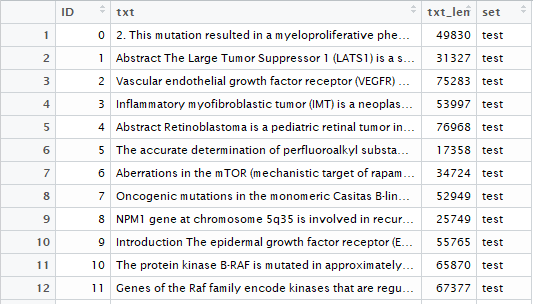
Feature Engineering

train\_txt



Initially the text file contains 2 columns(ID and text) and 3321 observations. I create two new columns namely, txt\_len and set.Now the train\_txt contain 4 variables and 3321 observations.

test\_txt

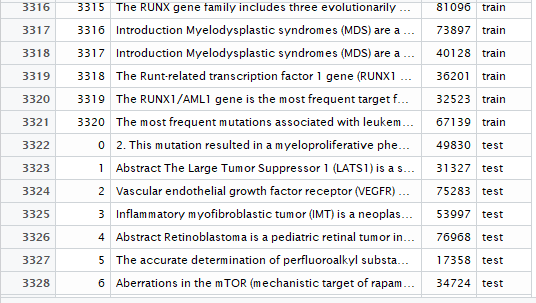


Initially the text file contains 2 columns(ID and text) and 5668 observations. After creating two new columns, txt\_len and set, the test\_text contain 4 variables and 5668 observations.

Merging train\_text and test\_text

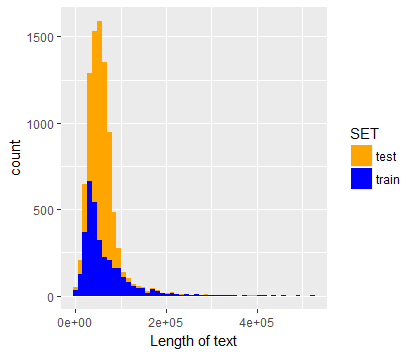
I merge the the train\_text and test\_text using full join, which in result give 8989 observations and 5 variables.

F:\shah\snapshots\combine_train_test_txt_variable_name.PNG



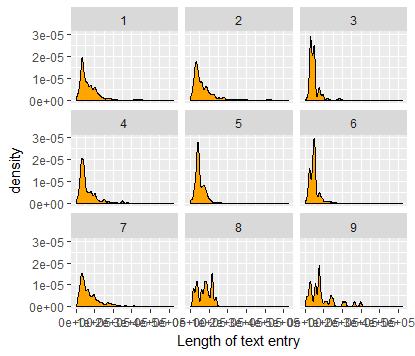
Now I want to look at the length distribution of the text features by using histogram plot.

For overall entries

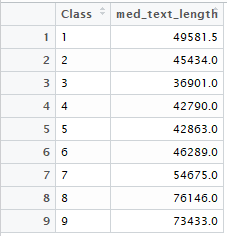


The difference in distribution plot might again be due to the machine-generated entries that have been added to the test sample.

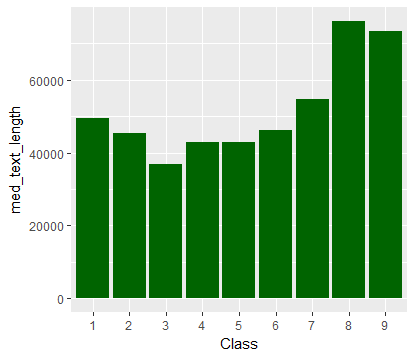
For different classes by facet wrap comparison



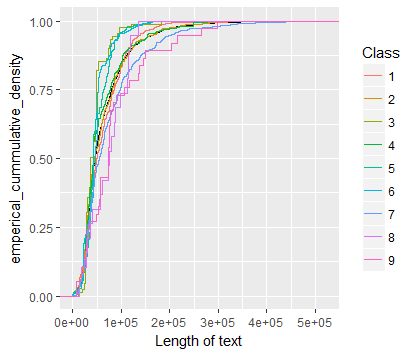
To get more clarity I want to plot bar graph for median text length for each class and also to plot empirical cumulative density functions(ecdf) for the same.



Bar plot for the median text length,



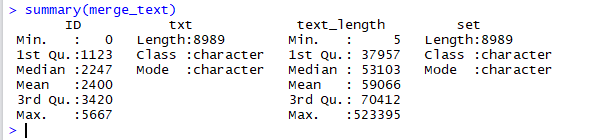
Empirical cumulative frequency distribution function (ecdf),



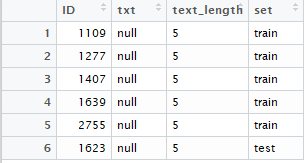
From the above calculations and plots we come to know that the shape and the median of the test length distribution are significantly different. Class 3 has the shortest or fewest papers associated with it. While Classes8 and are having text on the higher side.

### Missing text values

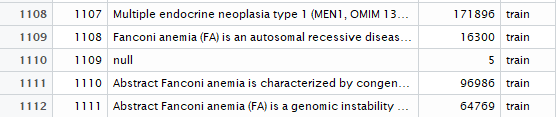
First I try to find the minimum length of text from the text length using summary function.

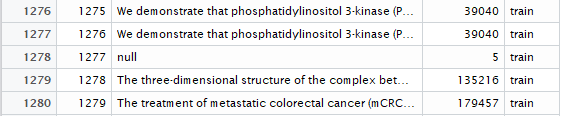


Here we can se the minimum length of text is 5.So I try to find all those strings that are shorter in length.



We can see there are 6 missing values, below I have displayed few of them.

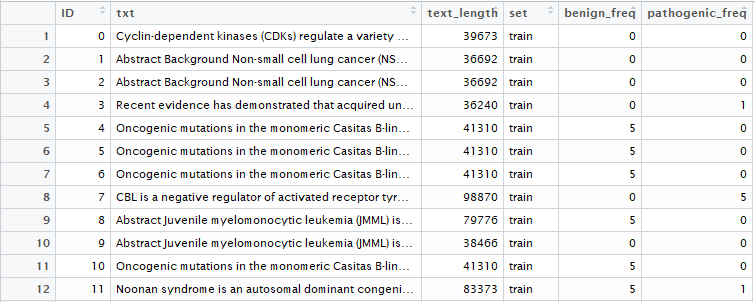




### Frequency of keywords - pedestrian approach

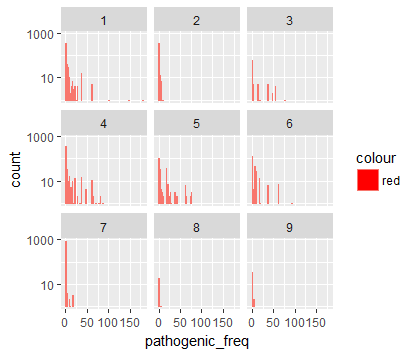
This approach based on standard string manipulation methods. Here the first step is to analyze the content of the clinical evidence to look how often certain keywords are mentioned in the text of the corresponding papers.

I find “pathogenic” and “benign” are the two words that are used frequently per observations.Now I try to extract the occurrences for those words.

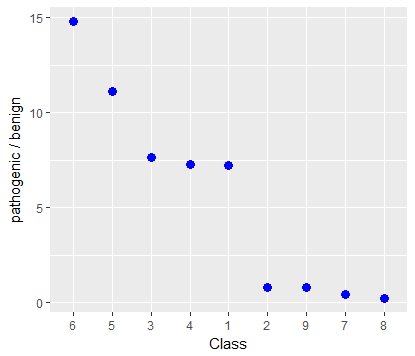


The frequency distributions of the word “pathogenic” for all the 9 classes. I have taken logarithmic scale to normalize the frequency on the y-axis.

“pathogenic” frequency distribution by using facet wrap function.



Now I want to  plot the ratio of the mean occurence per class of the word “pathogenic” over the mean occurence of the word “benign”.

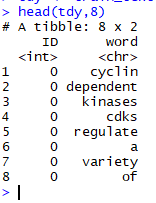


* The facet plot depicts that the Classes  1, 4, or 5 having the word “pathogenic” more frequent .
* The Classes 5 and 6 having higher ratio of “pathogenic” over “benign”.
* The ratio plot shows the two distinct groups of Classes: 2, 7, 8, 9 versus 1, 3, 4. The latter have on average a higher ratio of mentions of “pathogenic” over “benign” than the former.

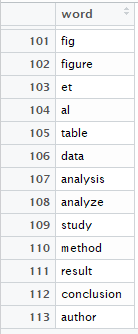
Text Analysis

Inspite of using different preprocessing steps to extract the token from the text I use tidytext package. Through tidy text we can use the powerful tools of the tidyverseto process and analyze text files. The tidytext format is being defined as a table with one token per row, a token being a word or another meaningful unit of text.

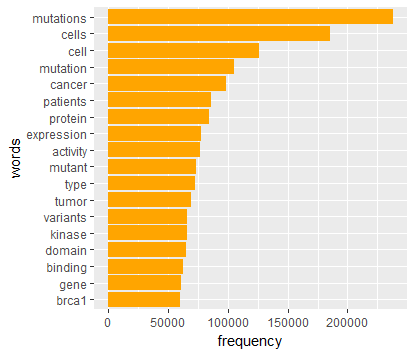
In order to get our text data in a tidy shape, we use the unnest\_tokens tool. This also removes punctuations and converts everything to lowercase:



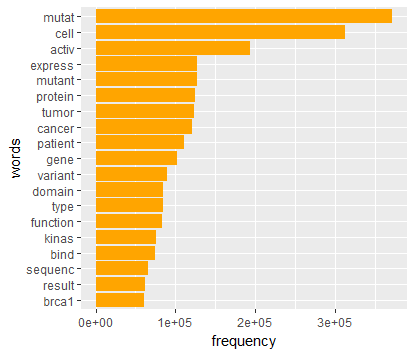
Now I create list of my own stopwords,



Plot for the most frequent tokens that we have extracted from the text,



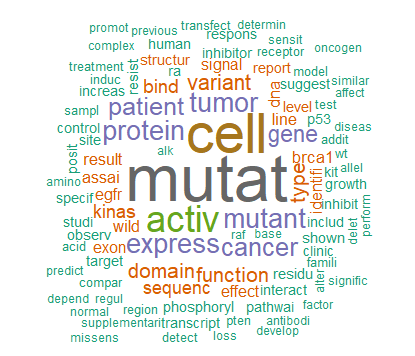
On the whole, those are words that we would expect to find in a publication on cancer research and genetics. We can observe the top 4 words are essentially 2 variants of two basic words each. Now I will reduce these variants into basic form using stemming tool.



The result conveys us the fundamental words that are most frequent in our overall text data.

Another method

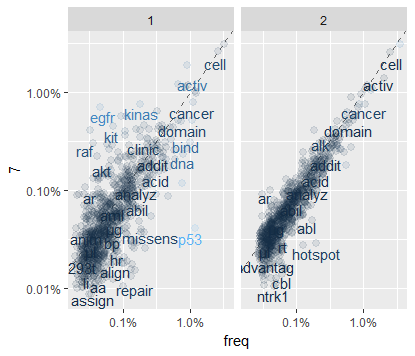
Word\_cloud



## Word frequencies depending on Classes

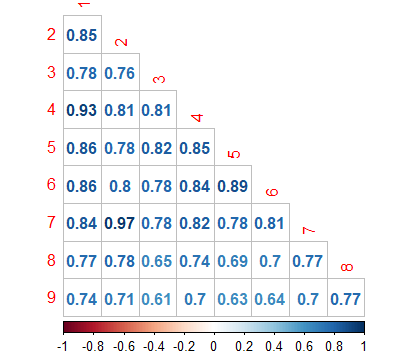
To use these word frequencies for prediction I first need to determine them for the individual Classes separately. Below, I join the “text” data with the Class information in the “variants” data set. Then after, I determine the relative frequency by Class of each word.

Now I will compare Class 7, the most frequent one, with Classes 1 and 2. Also, I will only look at words with more than 1000 occurrences per Class to get an idea.

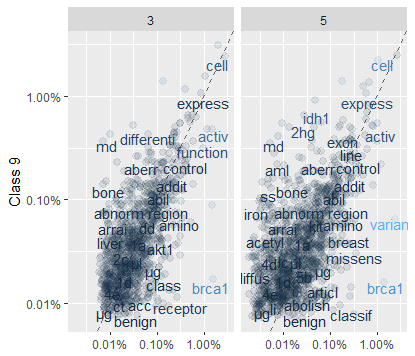


In the above plots, words that are close to the dashed line (having same frequency) have similar frequencies in the corresponding Classes. The words that are further along a particular Class axis (such as “inhibitor” for Class 7 versus 1) are more frequent in that Class. The blue-gray scale indicates how different the Class 7 frequency is from the overall frequency (with higher relative frequencies being lighter). The (slightly jittered) points in the background represent the complete set of (high-frequency) words, whereas the displayed words have been chosen to avoid overlap. By viewing the plot for class 1 and 7 we can get an idea that Classes2 and 7 are more similar than 1 and 7.

To get more clarity and information I will plot correlation matrix for the whole text data,



* Classes 2 and 7 are in fact the most similar ones here, followed by 1 and 4 having correlation coefficients above 0.90.
* Altogether, the most different Class appears to be number 9, in particular compared to classes 3 and 5. Now I want to see word frequency spread looks like for those combinations:



* These plots seems scatter than the previous set of plots ,particularly class 9 vs 5.
* benign” and “pathogen” are more frequent in Class 3 vs 9.

## TF-IDF analysis

## TF stands for term frequency; essentially how often a word appears in the text. We already measured above. A list of stop-words can be used to filter out frequent words.

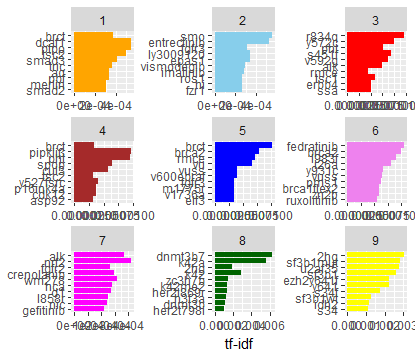
## IDF means inverse document frequency. Here, we give more emphasis to words that are rare within a collection of documents.

## TF-IDF, a heuristic index telling us how frequent a word is in a certain context(a certain Class) within the context of a larger document (all Classes).We can understand it as a normalization of the relative text frequency by the overall document frequency.

## F:\shah\personalized medicines\dirctry\most_characteristics_words.png

These are the most characteristics words associated with tf-idf values. We can see the top two yellow bars(of class 8)having the high tf-idf values, that means those words would really help in order to build train model.

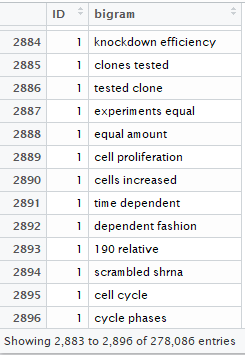
Now I want the most characteristics words with their frequencies for each class.



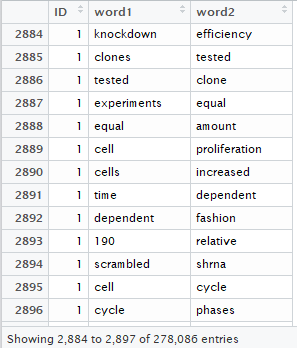
Word pair frequencies:

Now I want to check the characteristics of the groups of words that occur together (like,”text analysis”).This gives me an idea about the (typical) relationships between words in a certain document. I will do this by using the concept of n-gram(tool of tidytext), where n being the number of adjacent words we want to study as a group. First I am going to extract the words in pair(called bi-grams) having similar characteristics in someway.

Bi-gram,

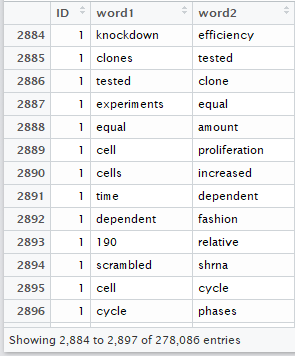


After saperating the bigram

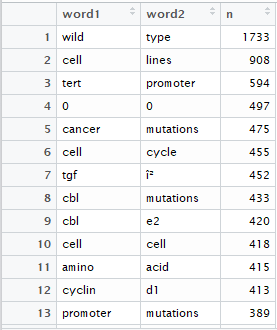


After that I filter out the stopwords by separate the bigrams first, and then later unite them back together.

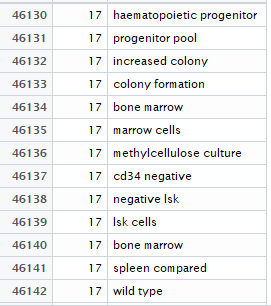
After removing stopwords,



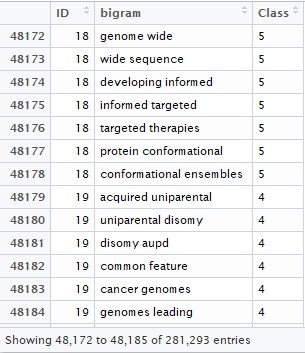
Bi-gram counting for each different pair,



Unite the bi-gram back after filtration,

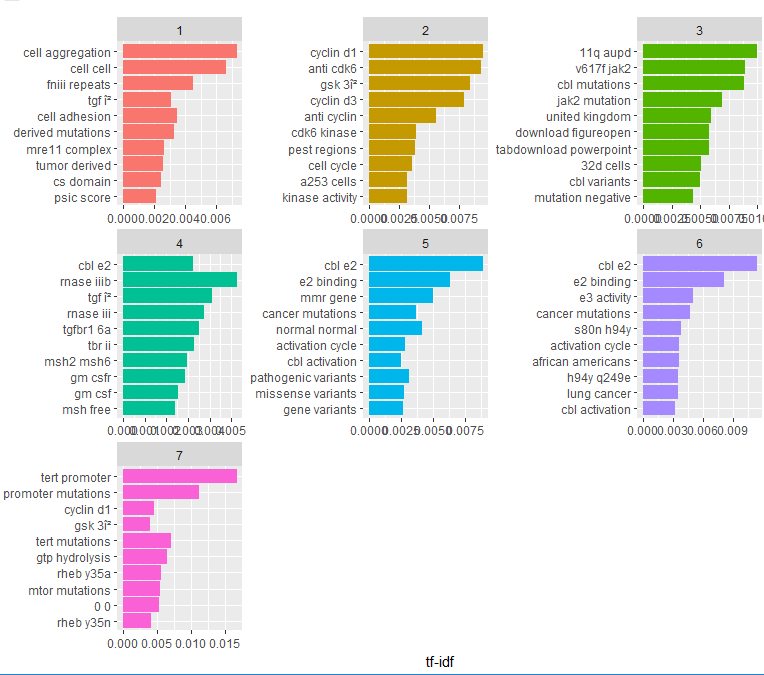


Adding the class for each pair of bi-gram,



The words in the above appeared in the pair in text document.

Bi-gram plot for the best tf-idf values.

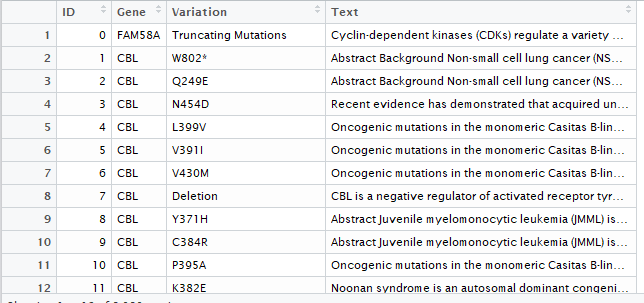


### Data Preparation:

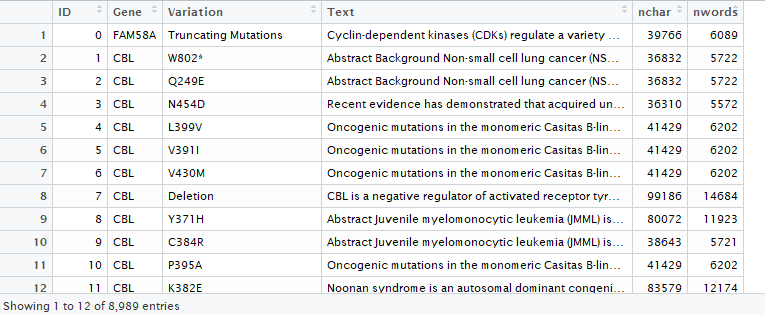
I have already explained about the data (csv and text) above and also pasted the detail description of the data in the form of snapshots and images. Before proceeding I further I want to explain briefly about the data.

Initially we have 4 data files, 2 csv and 2 text files. Here, Loading csv file into R is easy, but loading text files require bit effort. Because. In the original text file, variables (ID and text) are separated by the pipe (||) and reside in the same column. I separate these variables into two different column manually.

Then I merge train file (csv and text) by “ID” into one file “train\_data”, similarly test file(csv and text) by “ID” into ”test\_data”. After taking out the class variable of train\_data into a variable “train\_data\_class”, I delete the class variable from the train\_data. And then I merge “train\_data” and “test\_data” into data. This is the data from which I will extract all the features that will help me to build model for training data.



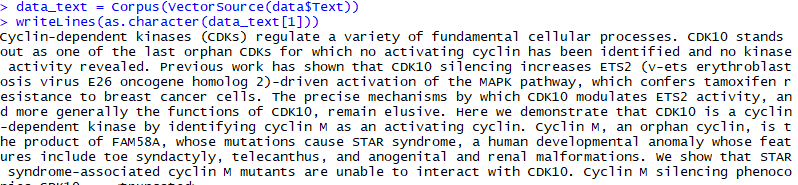
Now I create two new columns (nchar and nword) that will count the number of characters and number of words corresponding to each text. Here the data file contains 6 variables.



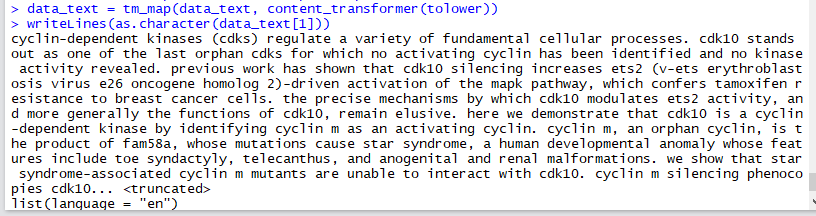
Now I start preprocessing for text data.

Text data Pre-processing :

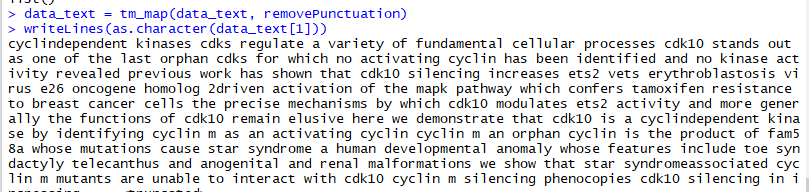
To start the preprocessing, first of all I convert text data into corpus.



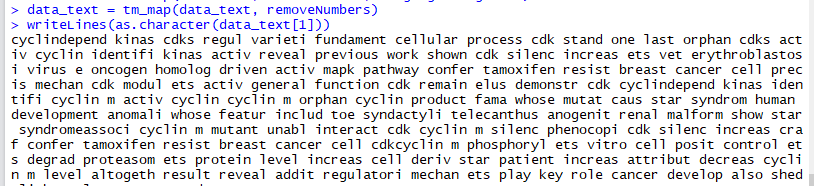
Most of the preprocessing steps are done by the “tm” library functionality. Removing all the unnecessary leading and trailing spaces by “stripWhitespace” is my step of pre-processing.Then I convert the text into lowercase by using “tolower” functionality.



After that I remove all the punctuation marks.

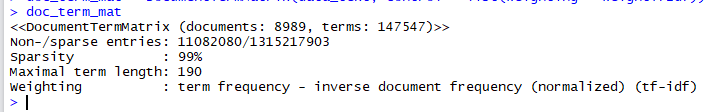


Then I remove stopwords from the whole document and apply root stemming(convert words to base/root form) as well. Then after I remove the digits from the text.



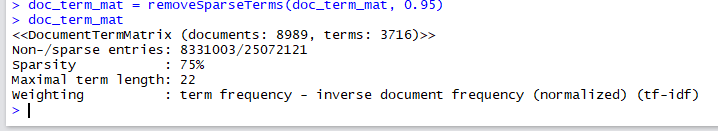
Term Document matrix

I create the term document matrix for the recently preprocessed text data.



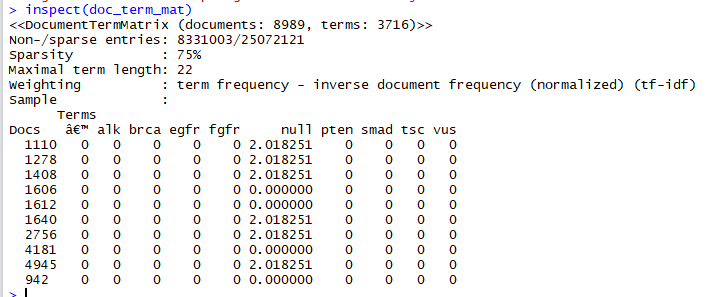
Here we can see plenty of sparse entries and the maximum length of any word also mentioned which 190.

Now I want to filter out these sparse entries.

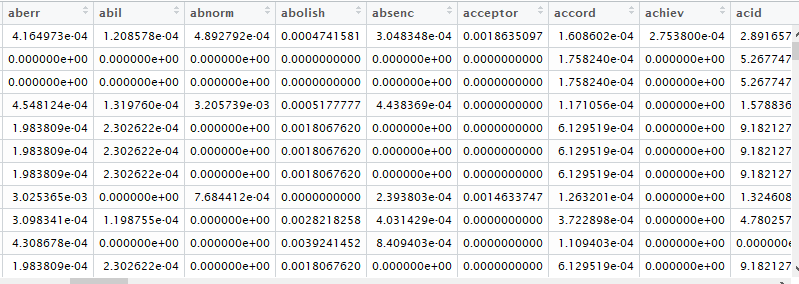


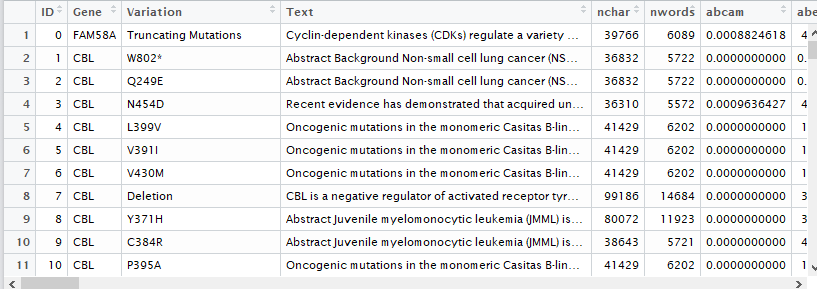
Here we can see the sparse intensity is reduced to 75% and the revised maximal length of any term is now 22.Also we can see the total number of terms reduced to 3716 from 147547.

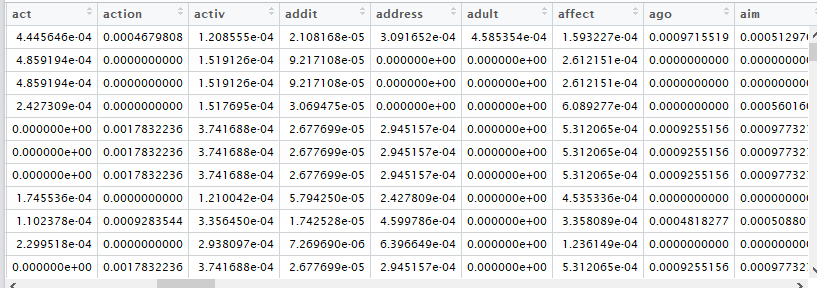
Inspection of document term matrix.



Now I join document term matrix into data, which in result produce 8989 observations and 3722 variables. Whose few rows and columns:







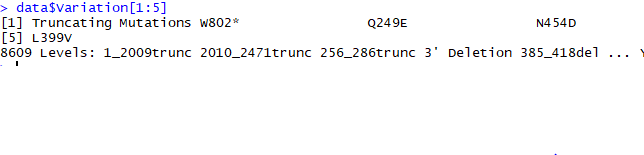
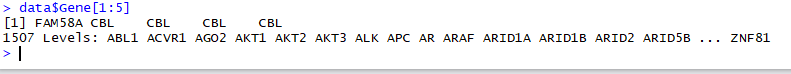
With this data I start to extract all the letters, numbers, symbols in the beginning, in between or at the end of variation for getting typical variation prediction with the help of “grepl”.

“grepl” function is used search for matches to argument pattern within each element of a character vector.

In this way I get 3785 variables with few new ones.

Then after I perform label count encoding for both Gene and variation variables.

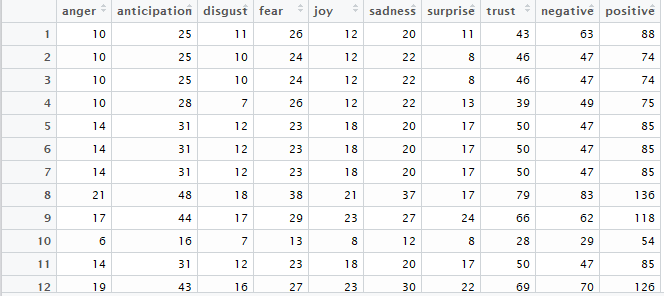
Before encoding,



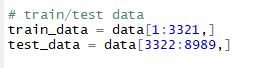
After encoding,

C:\Users\Ajaz\Desktop\snapssss\gene_after_label_count_encoding.PNGC:\Users\Ajaz\Desktop\snapssss\variation_after_label_count_encoding.PNG

Now I start sentiment analysis. First of all I extract sentiments of the text using get\_nrc\_sentiment() function. Then I combine these sentiment with the data.



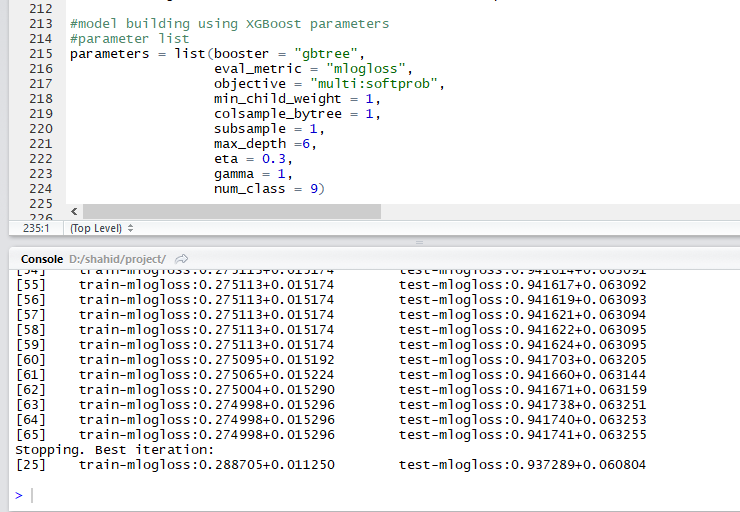
After this I move towards the splitting of data into train and test. As we know size of test data is larger than the train data for this problem. By keeping this thing in my mind I split train and test data in the ratio of approx.. 35% and 65% resp.



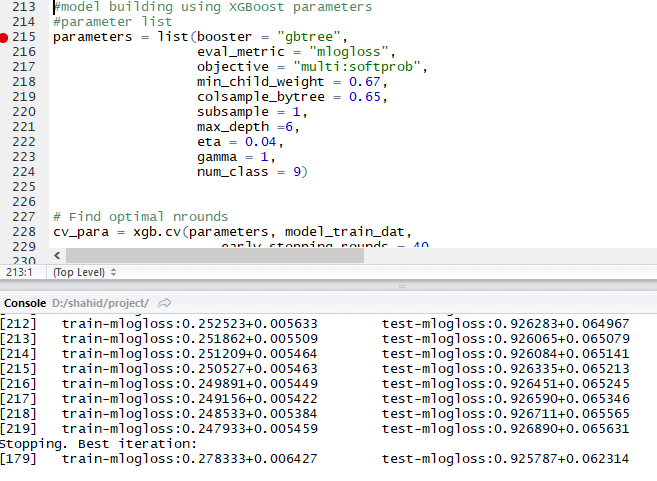
After splitting data into train and test, convert all the variables in numeric.

### Model Building

I start building my model with default using XGBoost algorithm and I get descent accuracy.



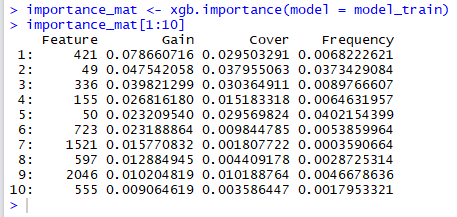
To improve the accuracy and optimal number of nrouns I tune the parameters using cross validation. In this I get this result,



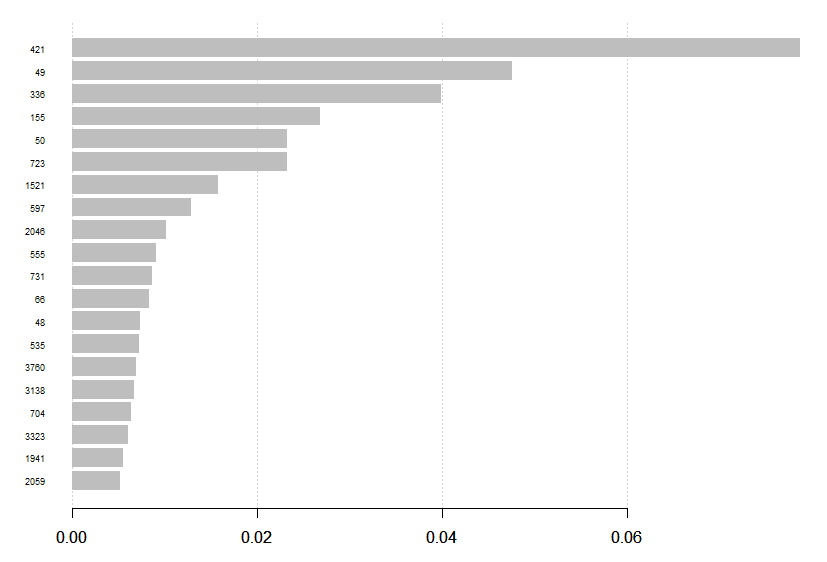
It clearly better accuracy than the default case. Here nround = 179.

Now I fix my nround to 179 and I get this result.

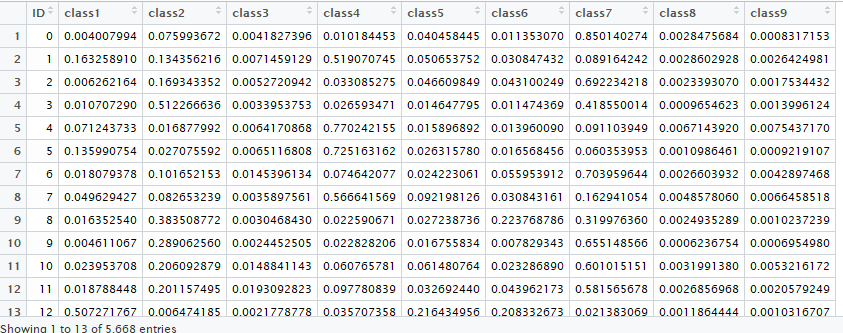
Importance of variables,



Plot for importance of variables,



**Prediction**

****

As I explained in the data exploration, class 7 is the most frequent one. Clearly it is having predominantly higher value, which means it helps the most in classifying the mutation.

### Result : Predicted id’s by the given classes have been generated as output. Overall accuracy of 73% has been achieved. Further investigation for more relevant features may be required to improve the accuracy.