ASSIGNMENT – TEXT MINING

1) Word Cloud – Amazon Dataset:

Clear the Ram and import the dataset:

```
> rm(list=ls())
> amazon=read.csv("amazon.csv")
```

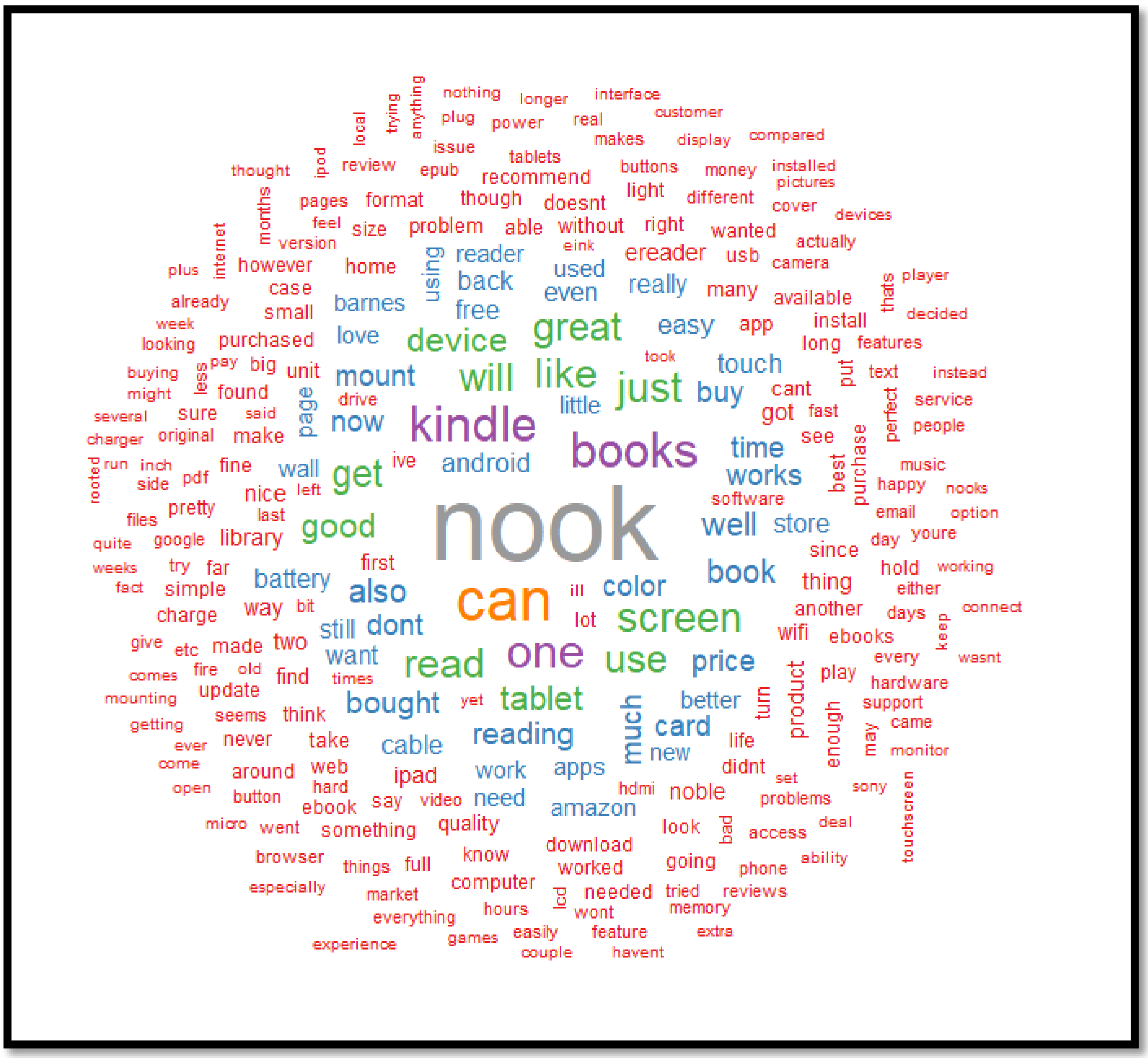
Data Cleaning and Re-organizing:

```
> regex_func=function(x){
+   return (gsub('[^a-z ]','',x))
+ }
> custom_stopwords=c()
> common_stopwords=stopwords()
> all_stop_words=c(custom_stopwords,common_stopwords)
> docs=as.character(amazon$reviewText)
> docs=VCorpus(VectorSource(docs))
> docs=tm_map(docs,content_transformer(tolower))
> docs=tm_map(docs,content_transformer(regex_func))
> docs=tm_map(docs,removeWords,all_stop_words)
> docs=tm_map(docs,stripWhitespace)
> dtm=DocumentTermMatrix(docs)
> df_dtm=as.data.frame(as.matrix(dtm))
> dim(df_dtm)
[1]  999 9322
> words_freq=colSums(df_dtm)
> library(data.table)
> words_freq=as.data.frame(words_freq)
> setDT(words_freq, keep.rownames = TRUE)[]
             rn words_freq
   1:        aaa          1
   2:        aac          1
   3: abandoning          1
  ---
9320:    zooming          3
9321:      zooms          1
9322:       zune          2

> colnames(words_freq)=c('words','frequency')
> library(dplyr)
> words_freq %>% arrange(-frequency) %>% head(10)
     words frequency
1     nook      1454
2      can       749
3    books       605
4   kindle       566
5      one       543
6   screen       475
7     just       460
8     like       453
9     read       434
10   great       422
```

ASSIGNMENT – TEXT MINING

Word-Cloud:

```
> library(wordcloud)
> wordcloud(words_freq$words,words_freq$frequency,min.freq = 50,
+           random.order = F,colors=brewer.pal(name='Set1',10))
Warning message:
In brewer.pal(name = "Set1", 10) :
  n too large, allowed maximum for palette Set1 is 9
Returning the palette you asked for with that many colors
```



ASSIGNMENT – TEXT MINING

2) Sentiment Analysis on Hotstar Review Dataset:

Data Cleaning and Re-organizing:

```
> library(tm)
> library(wordcloud)
> regex_func=function(x){
+   return (gsub('[^a-z #@]','',x))
+ }
> custom_stopwords=c()
> common_stopwords=stopwords()
> all_stop_words=c(custom_stopwords,common_stopwords)
> hotstar=read.csv("hotstar.csv")
> View(hotstar)
> dim(hotstar)
[1] 5053   13
> docs=as.character(hotstar$Reviews)
> docs=VCorpus(VectorSource(docs))
> docs=tm_map(docs,content_transformer(tolower))
> docs=tm_map(docs,content_transformer(regex_func))
> docs=tm_map(docs,removeWords,all_stop_words)
> docs=tm_map(docs,stripWhitespace)
> dtm=DocumentTermMatrix(docs)
> df_dtm=as.data.frame(as.matrix(dtm))
> dim(df_dtm)
[1] 5053 7107
> words_freq=colSums(df_dtm)
> library(data.table)
> words_freq=as.data.frame(words_freq)
> setDT(words_freq, keep.rownames = TRUE)[]
                       rn words_freq
   1:       #aadhaarmemes          2
   2: #aalaporaantamizhan          2
   3:            #aarambh          4
   4:          #abeergiri          2
   5:  #addictionproblems          1
  ---
7103:                zero          1
7104:               zinab          1
7105:            zindabad          2
7106:                znmd          1
7107:             zombies          1
> colnames(words_freq)=c('words','frequency')


> colnames(words_freq)=c('words','frequency')
> library(dplyr)
> top_words=words_freq %>% arrange(-frequency) %>% head(25)
> df_dtm_subset=df_dtm[,top_words$words]
> df_dtm_subset$sentiment=hotstar$Sentiment_Manual
> dim(df_dtm_subset)
[1] 5053   26
```

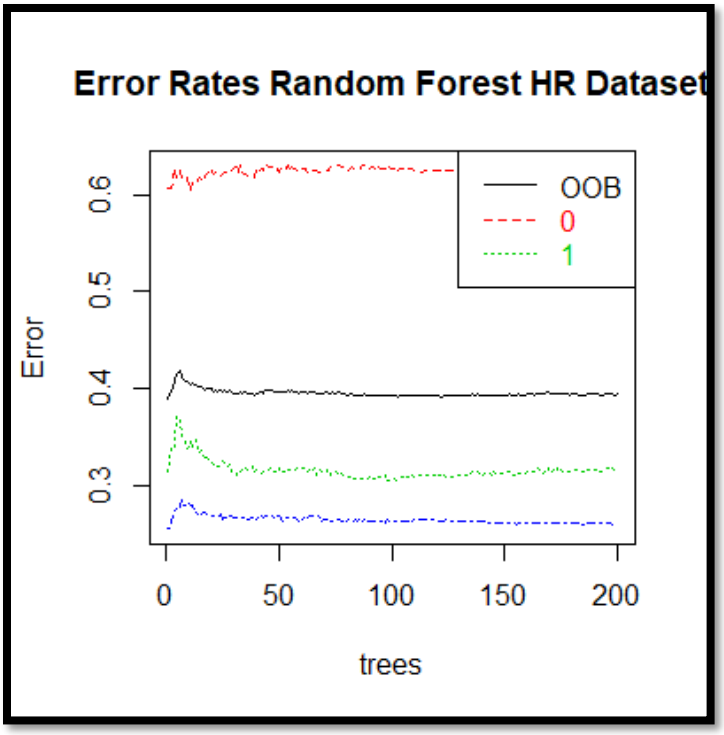
ASSIGNMENT – TEXT MINING

Splitting the Dataset into Test and Train:

```
> newdata=sample(1:nrow(df_dtm_subset),size=0.3*nrow(df_dtm_subset))
> test = df_dtm_subset[newdata, ]
> train = df_dtm_subset[-newdata,]
> dim(test)
[1] 1515   26
> dim(train)
[1] 3538   26
> attach(train)
```

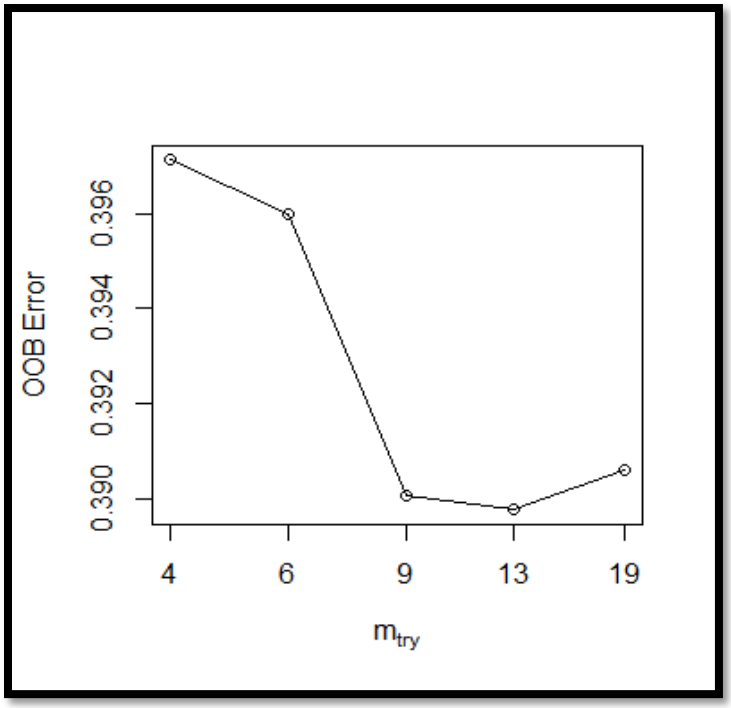
Random Forest Modeling:

```
> library(randomForest)
> RF <- randomForest(as.factor(sentiment) ~ ., data = train, ntree=200, mtry = 6,
nodesize = 100, importance=TRUE)
> print(RF)

Call:
 randomForest(formula = as.factor(sentiment) ~ ., data = train,      ntree = 200,
mtry = 6, nodesize = 100, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 200
No. of variables tried at each split: 6

        OOB estimate of  error rate: 39.4%
Confusion matrix:
         Negative Neutral Positive class.error
Negative      413     343      354   0.6279279
Neutral       123     839      263   0.3151020
Positive      108     203      892   0.2585204

> plot(RF, main="")
> legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)
> title(main="Error Rates Random Forest HR Dataset")
```



Fine Tuning the RF Model:

```
> tRF <- tuneRF(x = train[,-26], y=as.factor(train$sentiment), mtryStart = 6,
ntreeTry=100, stepFactor = 1.5,
+               improve = 0.0001, trace=TRUE, plot = TRUE, doBest = TRUE, nodesize =
100, importance=TRUE)
mtry = 6  OOB error = 39.6%
Searching left ...
mtry = 4         OOB error = 39.71%
-0.002855103 1e-04
Searching right ...
mtry = 9         OOB error = 39.01%
0.01498929 1e-04
mtry = 13        OOB error = 38.98%
0.0007246377 1e-04
mtry = 19        OOB error = 39.06%
-0.002175489 1e-04
```




```
> tRF

Call:
 randomForest(x = x, y = y, mtry = res[which.min(res[, 2]), 1],      nodesize = 100,
importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 13

        OOB estimate of  error rate: 38.84%
Confusion matrix:
         Negative Neutral Positive class.error
Negative      453     319      338   0.5918919
Neutral       142     833      250   0.3200000
Positive      123     202      878   0.2701579
```

Checking the Model accuracy with Development dataset:

```
> train$predict.class <- predict(tRF, train, type="class")
> train$predict.score <- predict(tRF, train, type="prob")
> with(train, table(sentiment, predict.class))

          predict.class
sentiment  Negative Neutral Positive
  Negative      486     305      319
  Neutral       121     857      247
  Positive      118     201      884
```

ASSIGNMENT – TEXT MINING

Checking the Model accuracy with Holdout dataset:

```
> test$predict.class <- predict(tRF, test, type="class")
> test$predict.score <- predict(tRF, test, type="prob")
> with(test, table(sentiment, predict.class))

          predict.class
sentiment  Negative Neutral Positive
  Negative      200     138      134
  Neutral        70     333      110
  Positive       68      77      385
```

Model Accuracy:

Train Dataset:

```
> (486+857+884)/(3538)
[1] 0.6294517 = 62.94 %
```

Test Dataset:

```
> (200+333+385)/(1515)
[1] 0.6059406 = 60.59 %
```