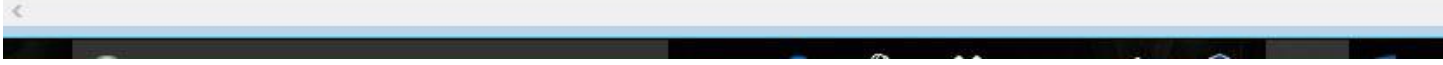
Problem-3

a) First, do the entire steps discussed in https://rpubs.com/pparacch/237109 to do naive Bayes classification on a dataset consisting of SMS messages. The data set on SMS messages is discussed at http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/ and can be downloaded from http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/smsspamcollection.zip

```
> rawData<-read.csv("G:/Fall Semester 2017/ISL/Assignment-2/output_SMSfile.txt",header=FALSE,stringsA
> colnames(rawData) <- c("type", "text")
> rawData$text <- iconv(rawData$text, to = "utf-8")
> rawData$type <- factor(rawData$type)
> a=1
> rep=100
> accuracy=dim(rep)
> precision=dim(rep)
> recall=dim(rep)
> for (k in 1:rep)
+ {
+   trainIndex <- createDataPartition(rawData$type, p = .8,
+                                     list = FALSE,
+                                     times = 1)
+   trainData <- rawData[trainIndex,]
+   testData <- rawData[-trainIndex,]
+   corpus <- Corpus(VectorSource(trainData$text))
+   corpus <- tm_map(corpus, content_transformer(tolower))
+ corpus <- tm_map(corpus, removeNumbers)
+ corpus <- tm_map(corpus, removeWords, stopwords())
+ corpus <- tm_map(corpus, removePunctuation)
+ corpus <- tm_map(corpus, stripWhitespace)
+   sms_dtm <- DocumentTermMatrix(corpus, control = list(global = c(2, Inf)))
+   sms_features <- findFreqTerms(sms_dtm, 5) #find words that appears at least 5 times
+   sms_dtm_train <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms_features))
```

```
+   sms_features <- findFreqTerms(sms_dtm, 5) #find words that appears at least 5 times
+   sms_dtm_train <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms_features))
+   convert_counts <- function(x){
+     x <- ifelse(x > 0, 1, 0)
+     x <- factor(x, levels = c(0,1), labels = c("No", "Yes"))
+     return (x)
+   }
+ sms_dtm_train <- apply(sms_dtm_train, MARGIN = 2, convert_counts)
+   sms_classifier <- naiveBayes(sms_dtm_train, trainData$type)
+   corpus <- Corpus(VectorSource(testData$text))
+   corpus <- tm_map(corpus, content_transformer(tolower))
+ corpus <- tm_map(corpus, removeNumbers)
+ corpus <- tm_map(corpus, removeWords, stopwords())
+ corpus <- tm_map(corpus, removePunctuation)
+ corpus <- tm_map(corpus, stripWhitespace)
+   sms_dtm_test <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms_features))
+   sms_dtm_test <- apply(sms_dtm_test, MARGIN = 2, convert_counts)
+   sms_test_pred <- predict(sms_classifier, sms_dtm_test)
+   tablin = table(testData$type, sms_test_pred)
+   accuracy[k] = (tablin[1,1]+tablin[2,2])/(sum(tablin))
+   precision[k] = (tablin[1,1])/(tablin[1,1]+tablin[2,1])
+   recall[k]=(tablin[1,1])/(tablin[1,1]+tablin[1,2])
+   if(a==1)
+   {
+     a=a+1
+     pal1 <- brewer.pal(9,"YlGn")
+     pal1 <- pal1[-(1:4)]
+
+     pal2 <- brewer.pal(9,"Reds")
+     pal2 <- pal2[-(1:4)]
+     par(mfrow = c(1,2))
+     wordcloud(corpus[trainData$type == "ham"], min.freq = 40, random.order = FALSE, colors = pal1
+     wordcloud(corpus[trainData$type == "spam"], min.freq = 40, random.order = FALSE, colors = pal
+   }
+ }
 +     pal2 <- brewer.pal(9,"Reds")
 +     pal2 <- pal2[-(1:4)]
 +     par(mfrow = c(1,2))
 +     wordcloud(corpus[trainData$type == "ham"], min.freq = 40, random.order = FALSE, colors = pal1)
 +     wordcloud(corpus[trainData$type == "spam"], min.freq = 40, random.order = FALSE, colors = pal2)
 +   }
 + }

There were 50 or more warnings (use warnings() to see the first 50)
> cat("Accuracy: ",mean(accuracy))
Accuracy:  0.8068386> cat("Precision :  ",mean(precision))
Precision :   0.8835561> cat("Recall: ",mean(recall))
Recall:  0.895117>
```

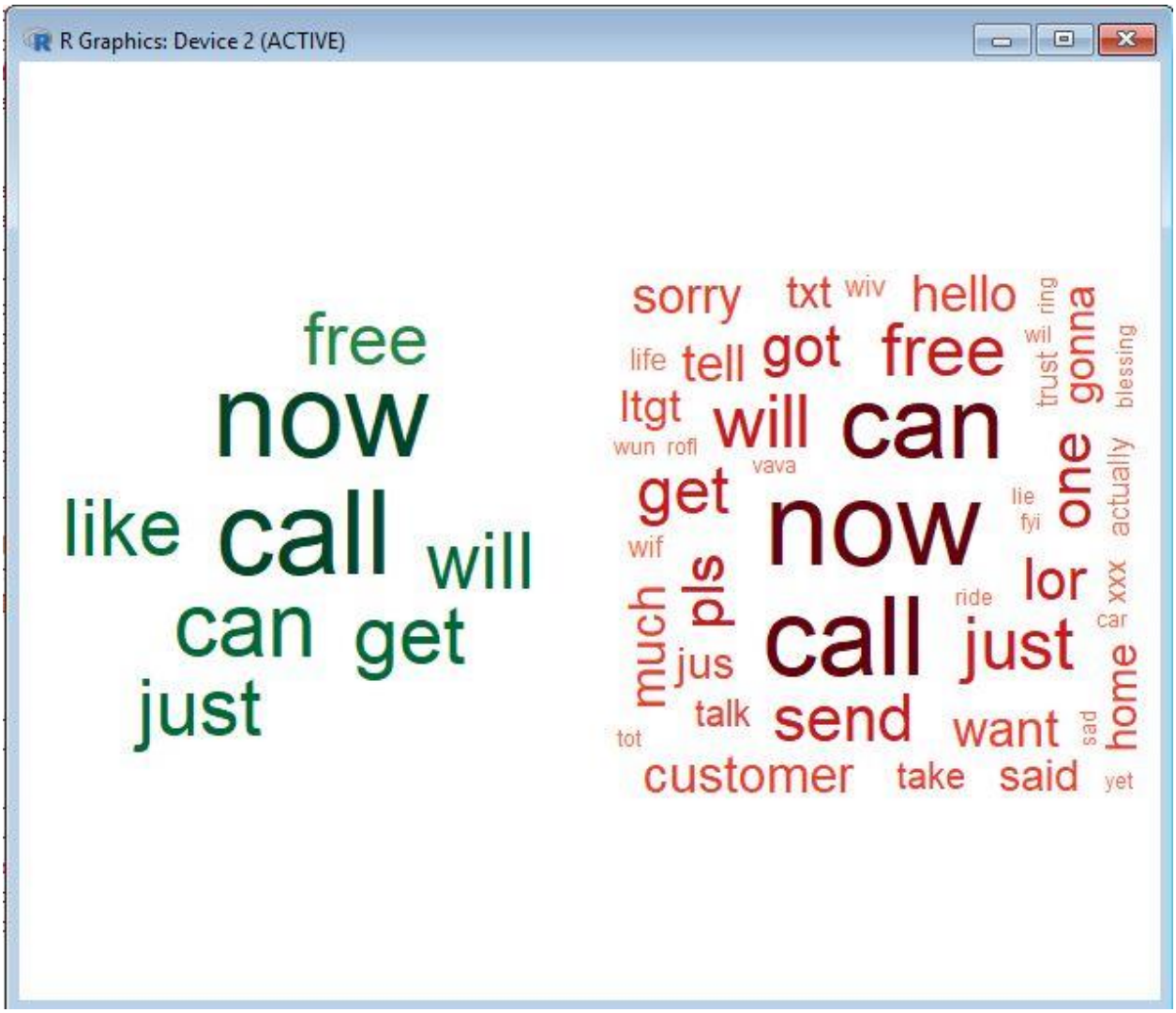


Accuracy: 0.806838

Precision: 0.8835561

Recall: 0.895117

Visual:



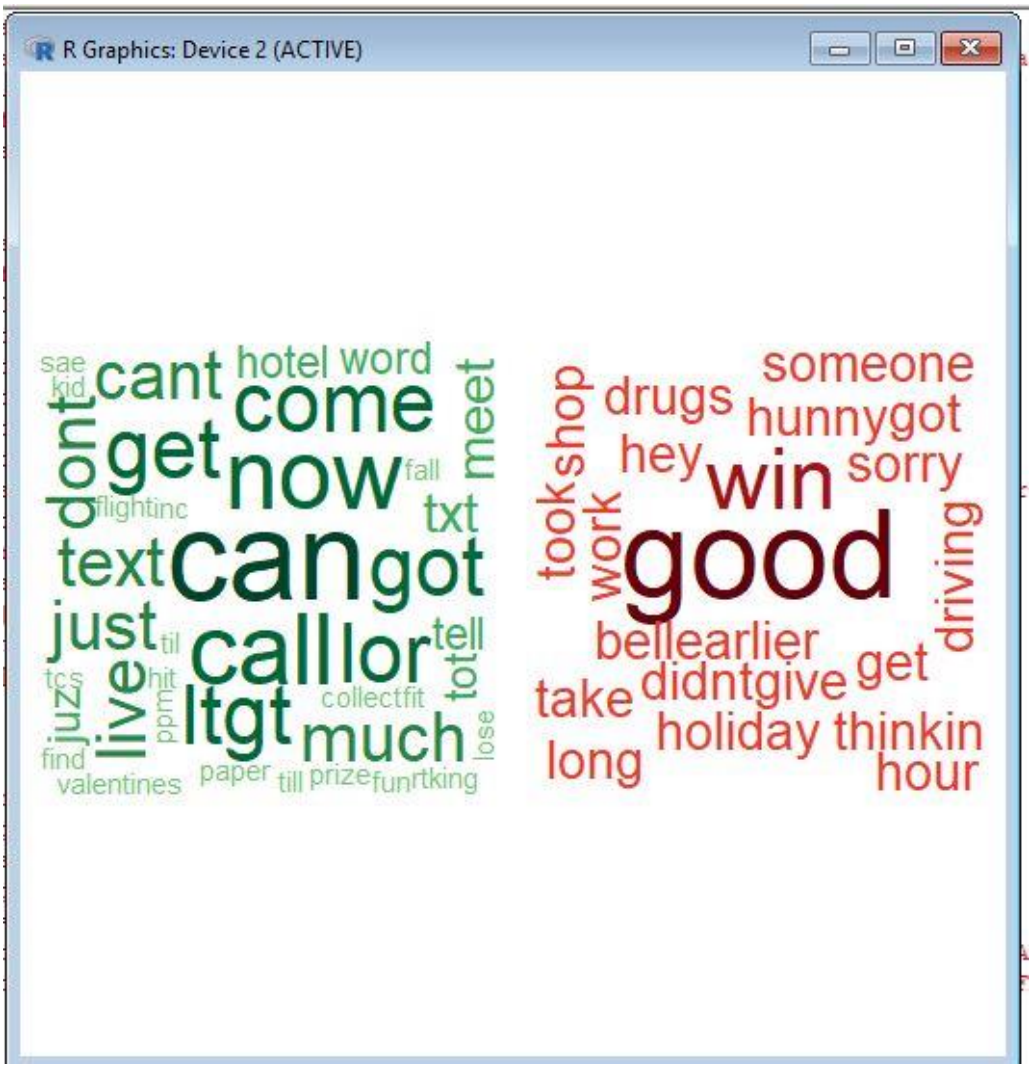
b) Now you are to consider a subset of 500 SMS messages from the original dataset using your last 4 digits of your student ID as the seed (set.seed(nnnn), where nnnn is the last 4 digits of your student ID) through sampling, using 'sample'. On this 500 SMS message in your collection, you'll then do 80/20-rule for training set/test data set split from YOUR data set. And repeat the above work performed in a) above. Report on how the results for

your set varies from the original dataset (be sure to include the wordcloud figure for your dataset alongside the original data set for visual comparison).

```
> rawData<-read.csv("G:/Fall Semester 2017/ISL/Assignment-2/output_SMSfile.txt",header=F,stringsAsFacto
> colnames(rawData)<-c("type","text")
> rawData$text<-iconv(rawData$text.to="utf-8")
Error: unexpected '=' in "rawData$text<-iconv(rawData$text.to="
> rawData$text<-iconv(rawData$text,to="utf-8")
> rawData$type <- factor(rawData$type)
> summary(rawData)
   type          text
 ham :4827   Length:5574
 spam: 747   Class :character
             Mode  :character
> a=1
> n=5574
> nt=500
> rep=100
> accuracy=dim(rep)
> precision=dim(rep)
> for(k in 1:rep)
+ {
+ set.seed(7910)
+ Index <- sample(1:n,nt)
+ rawData500 <- rawData[Index,]
+ trainIndex <- createDataPartition(rawData500$type, p = .8,list=FALSE,times=1)
+ trainData <- rawData500[trainIndex,]
+   testData <- rawData500[-trainIndex,]
+ corpus<-Corpus(VectorSource(trainData$text))
+ corpus <- tm_map(corpus, content_transformer(tolower))
+   corpus <- tm_map(corpus, removeNumbers)
+ corpus <- tm_map(corpus, removeWords, stopwords())
+ corpus <- tm_map(corpus, removePunctuation)
+ corpus <- tm_map(corpus, stripWhitespace)
+ sms_dtm <- DocumentTermMatrix(corpus, control = list(global = c(2, Inf)))
+ sms_features <- findFreqTerms(sms_dtm, 5)
```

```
- sms_dtm_train <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms_features))
- convert_counts <- function(x){
-     x <- ifelse(x > 0, 1, 0)
-     x <- factor(x, levels = c(0,1), labels = c("No", "Yes"))
-     return (x)
- }
- sms_dtm_train <- apply(sms_dtm_train, MARGIN = 2, convert_counts)
- sms_classifier <- naiveBayes(sms_dtm_train, trainData$type)
- corpus <- Corpus(VectorSource(testData$text))
- corpus <- tm_map(corpus, content_transformer(tolower))
- corpus <- tm_map(corpus, removeNumbers)
- corpus <- tm_map(corpus, removeWords, stopwords())
- corpus <- tm_map(corpus, removePunctuation)
- corpus <- tm_map(corpus, stripWhitespace)
- sms_dtm_test <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms_features))
-   sms_dtm_test <- apply(sms_dtm_test, MARGIN = 2, convert_counts)
- sms_test_pred <- predict(sms_classifier, sms_dtm_test)
- tablin = table(testData$type, sms_test_pred)
-   accuracy[k] = (tablin[1,1]+tablin[2,2])/(sum(tablin))
-   precision[k] = (tablin[1,1])/(tablin[1,1]+tablin[2,1])
-   recall[k]=(tablin[1,1])/(tablin[1,1]+tablin[1,2])
-   if(a==1)
```

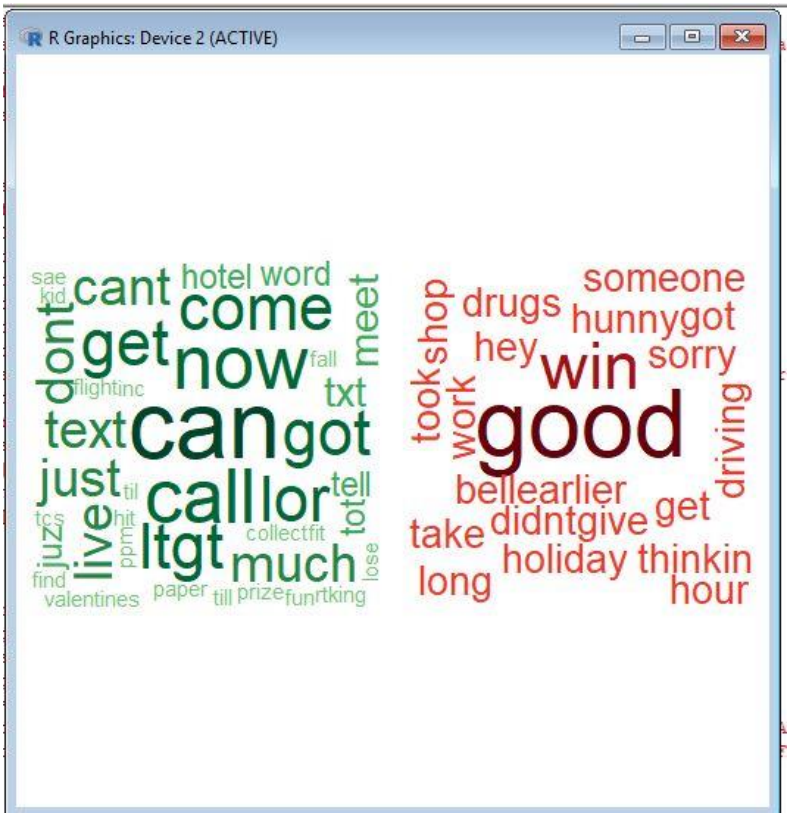
```
+   if(a==1)
+  {
+   a=a+1
+   pal1 <- brewer.pal(9,"YlGn")
+   pal1 <- pal1[-(1:4)]
+ pal2 <- brewer.pal(9,"Reds")
+   pal2 <- pal2[-(1:4)]
+ par(mfrow = c(1,2))
+   wordcloud(corpus[trainData$type == "ham"], min.freq = 40, random.order = FALSE, colors = pal1)
+   wordcloud(corpus[trainData$type == "spam"], min.freq = 40, random.order = FALSE, colors = pal2)
+ }
+ }
There were 50 or more warnings (use warnings() to see the first 50)
> cat("Accuracy: ",mean(accuracy))
Accuracy:  0.7575758> cat("Precision :  ",mean(precision))
Precision :   0.8554217> cat("Recall: ",mean(recall))
Recall:  0.8554217>
```



Accuracy: 0.7575758

Precision: 0.8554217, Recall: 0.8554217

Original Data Vs 500 Samples Data:



Summary and Inference:

1. From the values of accuracy, precision and recall we could infer that the full data set when replicated for 100 times and then 80% used as training and 20% used as testing gave better values than the sampled subset of 500 points.
2. I understood that based in size of data set the processing time varies.
3. The packages like caret, tm, word cloud are helpful in processing and visualising the text data.
4. The bigger presentation of the word in the visual represents high frequency of it being used in the text.