Project Part 2 Report

Yilun Yang

For logical type and numeric/integer type, we use different ways to test if the method is good or not.

We have known whether an email message is a HAM or a SPAM from the result of isSpam function. For other logical variables, we compare the results from them with true values from isSpam. Result is shown below:

```
$is.Re
FALSE  TRUE FALSE  TRUE
 2830  2034  1620    57

$replyUnderline
FALSE  TRUE FALSE  TRUE
 3515  1349   907   770

$isInReplyTo
FALSE  TRUE FALSE
 2909  1955  1677

$multipartText
FALSE  TRUE FALSE  TRUE
 4613   251  1384   293

$subjectPunctuationCheck
FALSE  TRUE FALSE  TRUE
 3558  1306  1317   360

$subjectSpamWords
FALSE  TRUE FALSE  TRUE
 4795    69  1503   174

$isOriginalMessage
FALSE  TRUE FALSE  TRUE
 4600   264  1676     1

$isDear
FALSE  TRUE FALSE  TRUE
 4836    28  1549   128

$isYelling
FALSE  TRUE FALSE  TRUE
 4857     7  1522   155

$priority
FALSE  TRUE FALSE  TRUE
 4861     3  1581    96
```

```
$is.Re
    FALSE       TRUE      FALSE       TRUE
```

```
0.58182566 0.41817434 0.96601073 0.03398927 

$replyUnderline
    FALSE      TRUE     FALSE      TRUE 
0.7226562 0.2773438 0.5408468 0.4591532 

$isInReplyTo
    FALSE      TRUE     FALSE 
0.5980674 0.4019326 1.0000000 

$multipartText
     FALSE       TRUE      FALSE       TRUE 
0.94839638 0.05160362 0.82528324 0.17471676 

$subjectPunctuationCheck
    FALSE      TRUE     FALSE      TRUE 
0.7314967 0.2685033 0.7853309 0.2146691 

$subjectSpamWords
     FALSE       TRUE      FALSE       TRUE 
0.98581414 0.01418586 0.89624329 0.10375671 

$isOriginalMessage
      FALSE        TRUE       FALSE        TRUE 
0.9457236842 0.0542763158 0.9994036971 0.0005963029 

$isDear
     FALSE       TRUE      FALSE       TRUE 
0.994243421 0.005756579 0.923673226 0.076326774 

$isYelling
     FALSE       TRUE      FALSE       TRUE 
0.998560855 0.001439145 0.907573047 0.092426953 

$priority
      FALSE        TRUE       FALSE        TRUE 
0.9993832237 0.0006167763 0.9427549195 0.0572450805 
```

The first table is the TRUE/FALSE numbers in HAM and SPAM datasets for all derived variables. The second table is the TRUE/FALSE ratio in HAM and SPAM datasets for all derived variables.

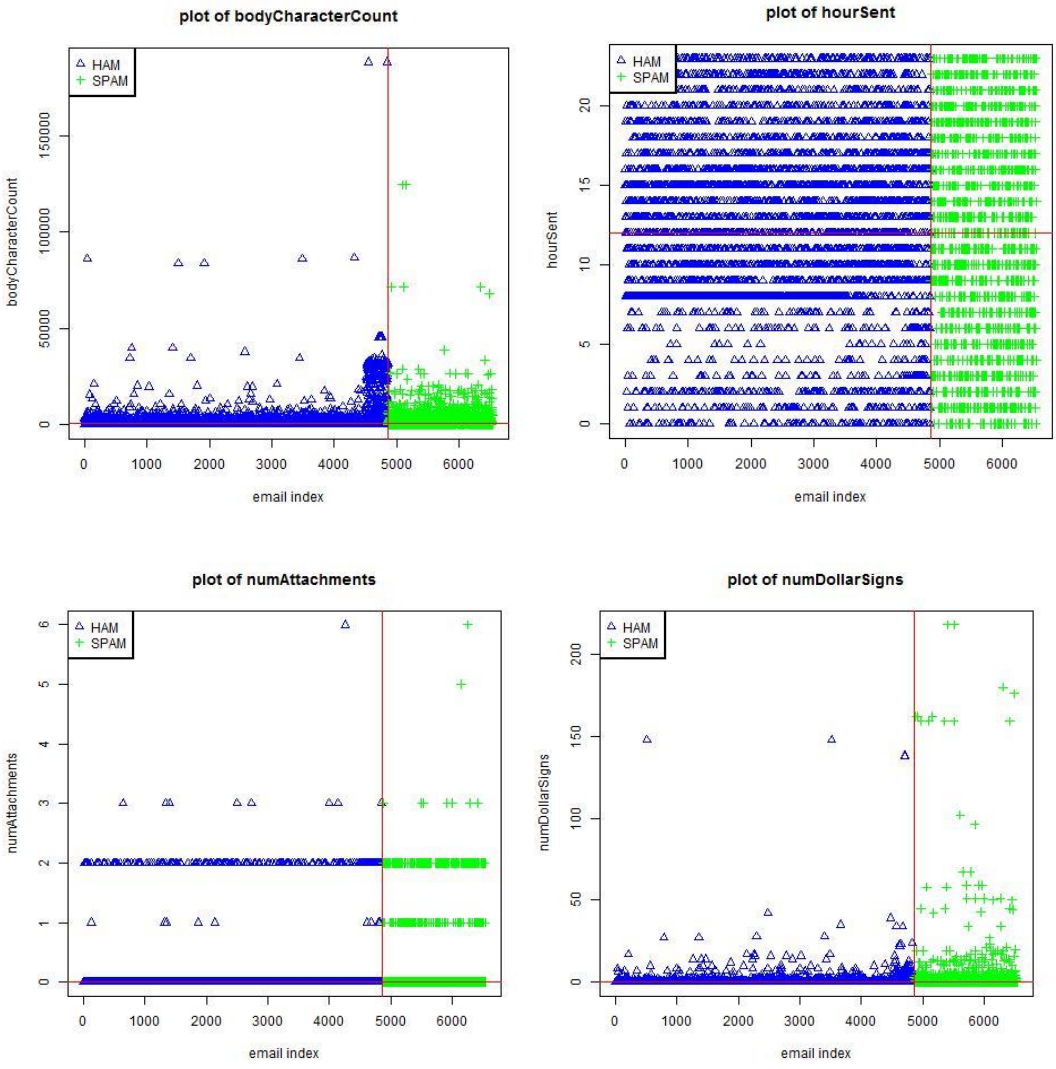
If a derived variable works well in classification, the ratio of truth or false in HAM and SPAM should be big enough to discriminate. In this way, we can check if each variable

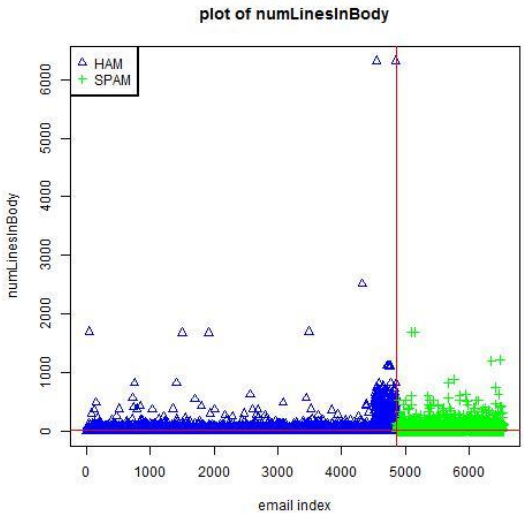
is significant in determining spam and ham.

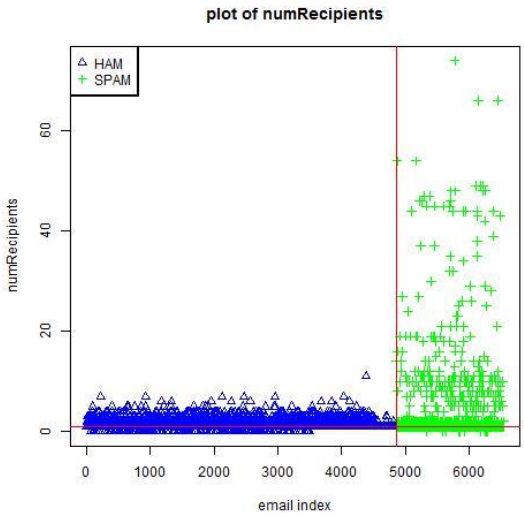
is.Re is a perfect variable, the ratio of truth in ham is more than 10 times than that in ham. So if we have an email is.Re is true, it is more likely to be a ham. However, since ratio of false are similar in both ham and spam, if we have an email is.Re is false, it's not easy to determine.

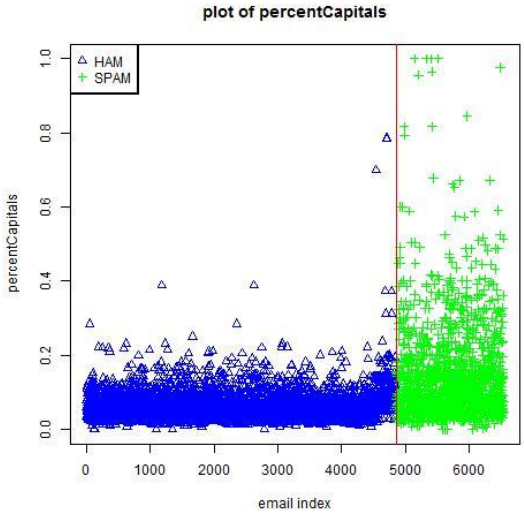
This rule can be applied to isInReplyTo, subjectSpamWords, isOriginalMessages, isDear, isYelling, priority. They are all good variables to discriminate spams but generally they are only good at determine emails with TRUE values. However, there are still some variables have false ratios in HAM and SPAM quite different like is.Re and isInReplyTo. Thus, I think these two derived variables are the best.

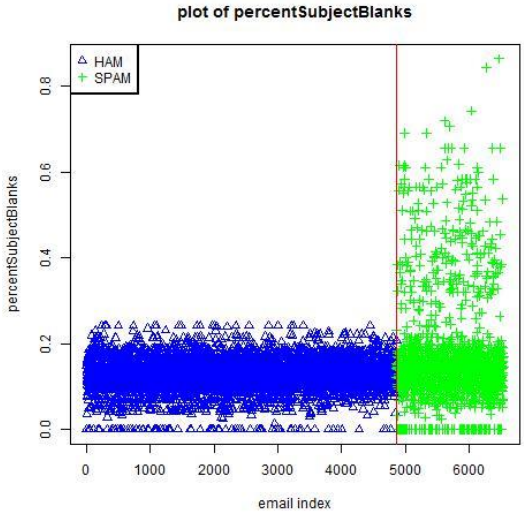
For numeric/integer variables, it's not easy to compare them with true values directly, I will plot scatterplots to illustrate. Plots are the following ones.

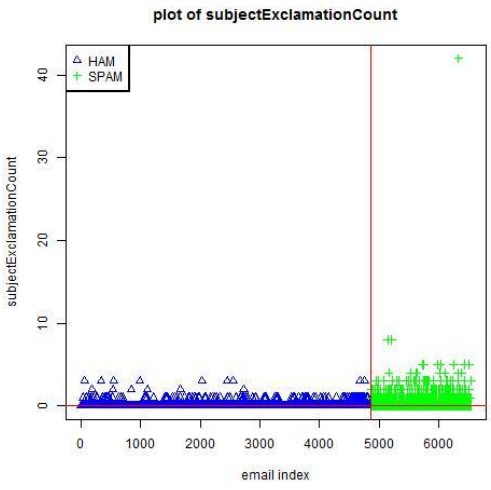


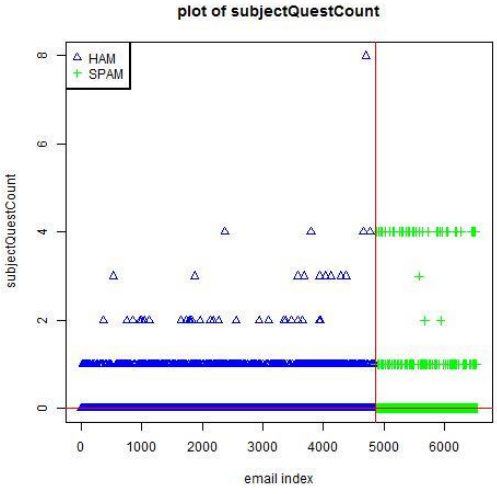












For our ten variables above, blue triangle sign represents the HAM emails and green plus sign represents SPAM emails. Horizontal line is the median line of all HAM emails, it helps us determine whether there is a significant difference between HAM emails and

SPAM emails when they are at the same level in a variable. The vertical line is the separation line of HAM and SPAM emails.

In these plots, bodyCharacterCount, numAttachments, numLinesInBody, numDollarSigns, subjectExclamationCount, subjectQuestCount are useless in determining spam email. hourSent, percentCapitals, percentSubjectBlanks have some utilities to determine spam emails but they are not the best method. For example, hourSent shows if an email is sent during 3am to 7am, it is more likely to be a spam email. However, if an email is sent in other times, this classification is no more applicable. This is the same for the latter two variables. When an email has percentCapitals or percentSubjectBlanks more than 0.2, it is much safer to classify this as a spam email but they are useless when the variables are less than 0.2.

The best one among all the given methods is numRecipients, the overlapped area of HAM and SPAM in this method is the smallest. We can get a frequency table of HAM like following if we delete the outlier (No. 4385).

0	1	2	3	4	5	6	7
190	3458	849	238	92	22	8	6

1 is both the median and mode of HAM emails, the percentage of SPAM that has more than 1 recipient is 0.234347(the percentage for 2 and 3 recipients are 0.1586166 and 0.1526535). Though they are not such big numbers that can safely discriminate spam emails, they are already the highest among all the given methods.

#Code:

```
##This for classify email messages into HAM or SPAM

##First we list the code of 20 methods we use
##use stringr library
library(stringr)

##tr                    trainMessages list

is.Spam <- function(tr){
  name <- names(tr)
  spam <- grepl("spam", name)

  spam
```

```
}

is.Re <- function(tr){
   isre <- vector(length = length(tr))
   #Get our the emails including subject
   subjind <- unlist(sapply(tr, function(x) "Subject" %in% names(x$header)))
   subject <- unlist(sapply(tr[subjind], function(x) x$header["Subject"]))

   #If Re: at the beginning of subject
   isre[subjind] <- ifelse(regexpr("Re:", subject) == 1, TRUE, FALSE)
   isre[!subjind] <- FALSE
   isre

}

numLinesInBody <- function(tr){
   numbody <- unlist(sapply(tr, function(x) length(x$body)))
   names(numbody) <- NULL
   numbody
}

bodyCharacterCount <- function(tr){
   numchar<-unlist(sapply(tr, function(x) sum(nchar(x$body, type = "bytes"))))
   names(numchar) <- NULL
   numchar
}

replyUnderline <- function(tr){
   replyind <- unlist(sapply(tr, function(x) any("Reply-To" == names(x$header))))
   names(replyind) <- NULL
   underline <- unlist(sapply(tr[replyind], function(x) grepl("[_[:alnum:]]",
x$header["Reply-To"])))
   names(underline) <- NULL
   replyind[replyind] <- underline

   replyind


}

subjectExclamationCount <- function(tr){
   countexclaim <- vector(length = length(tr))
   subjind <- unlist(sapply(tr, function(x) "Subject" %in% names(x$header)))
   subject <- unlist(sapply(tr[subjind], function(x) x$header["Subject"]))
```

```
    countexclaim[subjind] <- str_count(subject, "!")
    countexclaim[!subjind] <- 0

    countexclaim
}

subjectQuestCount <- function(tr){
    countques <- vector(length = length(tr))
    subjind <- unlist(sapply(tr, function(x) "Subject" %in% names(x$header)))
    subject <- unlist(sapply(tr[subjind], function(x) x$header["Subject"]))
    countques[subjind] <- str_count(subject, "\\?")
    countques[!subjind] <- 0

    countques
}

numAttachments <- function(tr){
    countattach <- unlist(sapply(tr, function(x) length(x$attachment)))
    names(countattach) <- NULL

    countattach
}

percentCapitals <- function(tr){
    percent <- vector(length = length(tr))
    bodyind <- sapply(tr, function(x) length(x$body) != 0)
    body <- sapply(tr, function(x) x$body)
    uplower <- unlist(sapply(body[bodyind], function(x) sum(str_count(x, "[a-zA-Z]"))))
    onlyupper <- unlist(sapply(body[bodyind], function(x) sum(str_count(x, "[A-Z]"))))
    percent[bodyind] <- onlyupper/uplower

    #Replace with NAN
    percent[!bodyind] <- 0/0

    percent
}

isInReplyTo <- function(tr){
    isReto <- sapply(tr, function(x) "In-Reply-To" %in% names(x$header))

    isReto
}

hourSent <- function(tr){
```

```
    time <- vector(length = length(tr))
    Dateind <- sapply(tr, function(x) "Date" %in% names(x$header))
    Date <- unlist(sapply(tr[Dateind], function(x) x$header["Date"]))
    matched <- regexpr("[0-9]{1,2}:", Date)

    #Extract the matched string and substitute ":" with " "
    hoursent <- gsub(":", "", regmatches(Date, matched))
    time[Dateind] <- hoursent
    time[!Dateind] <- 00

    as.numeric(time)
}

multipartText <- function(tr){
    countmutext <- vector(length = length(tr))
    contypeind <- unlist(sapply(tr, function(x) "Content-Type" %in% names(x$header)))
    contenttype <- unlist(sapply(tr[contypeind], function(x) x$header["Content-Type"]))

    #There is no match for multipart/text, I match with only multipart
    countmutext[contypeind] <- grepl("[mM]ultipart", contenttype)
    countmutext[!contypeind] <- FALSE

    countmutext
}

subjectPunctuationCheck <- function(tr){
    puncheck <- vector(length = length(tr))
    subjind <- unlist(sapply(tr, function(x) "Subject" %in% names(x$header)))
    subject <- unlist(sapply(tr[subjind], function(x) x$header["Subject"]))
    puncheck[subjind] <- grepl("[A-Za-z][0-9[:punct:]]+[A-Za-z]", subject)
    puncheck[!subjind] <- FALSE

    puncheck
}

subjectSpamWords <- function(tr){
    Spamword <- vector(length = length(tr))
    subjind <- unlist(sapply(tr, function(x) "Subject" %in% names(x$header)))
    subject <- unlist(sapply(tr[subjind], function(x) x$header["Subject"]))
    Spamword[subjind] <-
grepl("([Vv]iagra|[Pp]ounds|[Ff]ree|[Ww]eight|[Gg]uarantee|[Mm]illions|[Dd]ollars|[
Cc]redit|[Rr]isk|[Pp]rescription|[Gg]eneric|[Dd]rug|[Mm]oney [Bb]ack|[Cc]redit
[Cc]ard)", subject)
    Spamword[!subjind] <- FALSE
```

```
    Spamword

}

percentSubjectBlanks <- function(tr){
    percent <- vector(length = length(tr))
    subjind <- unlist(sapply(tr, function(x) "Subject" %in% names(x$header)))
    subject <- unlist(sapply(tr[subjind], function(x) x$header["Subject"]))
    countblank <- str_count(subject, " ")
    countall <- str_count(subject, ".")
    percent[subjind] <- countblank/countall
    percent[!subjind] <- 0/0

    percent
}

isOriginalMessage <- function(tr){
    isorigin <- vector(length = length(tr))
    bodyind <- sapply(tr, function(x) length(x$body) != 0)
    body <- sapply(tr, function(x) x$body)
    isorigin[bodyind] <- unlist(sapply(body[bodyind], function(x)
any(grepl("[Oo]riginal [Mm]essage", x))))
    isorigin[!bodyind] <- FALSE

    isorigin
}

numDollarSigns <- function(tr){
    numsign <- vector(length = length(tr))
    bodyind <- sapply(tr,function(x) length(x$body) != 0)
    body <- sapply(tr, function(x) x$body)
    numsign[bodyind] <- sapply(body[bodyind], function(x) sum(str_count(x, "\\$")))
    numsign[!bodyind] <- 0

    numsign
}

isDear <- function(tr){
    startdear <- vector(length = length(tr))
    bodyind <- sapply(tr, function(x) length(x$body) != 0)
    body <- sapply(tr, function(x) x$body)
    startdear[bodyind] <- unlist(sapply(body[bodyind], function(x)
any(regexpr("[Dd](EAR|ear)", x) == 1)))
```

```
  startdear[!bodyind] <- FALSE

  startdear
}

isYelling <- function(tr){
  capsub <- vector(length = length(tr))
  subjind <- unlist(sapply(tr, function(x) "Subject" %in% names(x$header)))
  subject <- unlist(sapply(tr[subjind], function(x) x$header["Subject"]))
  capsub[subjind] <- ifelse(str_count(subject, "[A-Z]") == str_count(subject, "[a-zA-
Z]"), TRUE, FALSE)
  capsub[!subjind] <- FALSE

  capsub
}

priority <- function(tr){
  highprior <- vector(length = length(tr))
  highprior1 <- vector(length = length(tr))
  highprior2 <- vector(length = length(tr))

  #Process X-Priority cases and put into a vector
  xpriorind <- unlist(sapply(tr, function(x) "X-Priority" %in% names(x$header)))
  xpriority <- unlist(sapply(tr[xpriorind], function(x) x$header["X-Priority"]))

  #both 1 and 2 are high priority
  highprior1[xpriorind] <- grepl("[12]", xpriority)

  #Process X-Msmail-Priority cases and put into a vector
  xmspriorind <- unlist(sapply(tr, function(x) "X-Msmail-Priority" %in%
names(x$header)))
  xmspriority <- unlist(sapply(tr[xmspriorind], function(x) x$header["X-Msmail-
Priority"]))
  highprior2[xmspriorind] <- grepl("[Hh](IGH|igh)", xmspriority)

  #priority is high for either of the them is high
  highprior <- highprior1|highprior2

  highprior
}

numRecipients <- function(tr){
  countnum <- vector(length = length(tr))
```

```
  #Get out "To" numbers
  Toind <- unlist(sapply(tr, function(x) "To" %in% names(x$header)))
  Tomail <- unlist(sapply(tr[Toind], function(x) x$header["To"]))
  countnum[Toind] <- str_count(Tomail, ",") + 1
  countnum[!Toind] <- 0

  #Get out "Cc" numbers
  ccind <- unlist(sapply(tr, function(x) any(grepl("^[Cc]{2}$", names(x$header)))))
  ccmail <- sapply(tr[ccind], function(x) x$header[(names(x$header) == "Cc" |
names(x$header) == "CC")])
  num <- unlist(sapply(ccmail, function(x) sum(str_count(x, ",")) + 1))
  countnum[ccind] <- countnum[ccind] + num

  countnum

}

#Next, we process all the data into a data frame with 20 columns of our classifiers
classdf <- function(tr){
  df <- data.frame(is.Re = is.Re(tr), numLinesInBody = numLinesInBody(tr),
                   bodyCharacterCount = bodyCharacterCount(tr),
replyUnderline = replyUnderline(tr),
                   subjectExclamationCount = subjectExclamationCount(tr),
subjectQuestCount = subjectQuestCount(tr),
                   numAttachments = numAttachments(tr), percentCapitals =
percentCapitals(tr),
                   isInReplyTo = isInReplyTo(tr), hourSent = hourSent(tr),
multipartText = multipartText(tr),
                   subjectPunctuationCheck = subjectPunctuationCheck(tr),
subjectSpamWords = subjectSpamWords(tr),
                   percentSubjectBlanks = percentSubjectBlanks(tr),
isOriginalMessage = isOriginalMessage(tr),
                   numDollarSigns = numDollarSigns(tr), isDear = isDear(tr),
isYelling = isYelling(tr), priority = priority(tr),
                   numRecipients = numRecipients(tr), row.names = NULL)
  df

}

##############################################################################
##################################################
##Test whether variables work well

df<-classdf(trainMessages)
```

```
## for logical type
   TRUEclass <- is.Spam(trainMessages)
   nspam <- sum(TRUEclass)
   nham <- sum(!TRUEclass)

   islogic <- sapply(df[1,], is.logical)
   dflogic <- df[,islogic]
   dfnum <- df[,!islogic]

   logictable <- sapply(dflogic, function(x) c(table(x[1 : nham]), table(x[(nham + 1) :
(nham + nspam)])))
   ratiotable <- sapply(dflogic, function(x) c(table(x[1 : nham])/nham, table(x[(nham +
1) : (nham + nspam)])/nspam))

## for numeric type


   for(i in names(dfnum)){
      jpeg(file = paste(i, ".jpg", sep = ""))
      plot(dfnum[[i]], pch = c(rep(2, nham), rep(3, nspam)),col = c(rep("blue", nham),
rep("green", nspam)),
      main = paste("plot of", i, sep = " "), ylab = i, xlab = "email index")
      abline(h = median(dfnum[[i]][1 : nham]), v = nham, col = c("red", "red"))
      legend("topleft", legend = c("HAM", "SPAM"), pch = c(2,3),col = c("blue",
"green"), box.lwd=2)
      dev.off()
   }
```