Using Statistics to Identify Spam

Anatomy of an email Message

Spam Data

```
head(list.files(path = file.path(data.dir, "easy_ham")))
```

```
[1] "00001.7c53336b37003a9286aba55d2945844c"
[2] "00002.9c4069e25e1ef370c078db7ee85ff9ac"
[3] "00003.860e3c3cee1b42ead714c5c874fe25f7"
[4] "00004.864220c5b6930b209cc287c361c99af1"
[5] "00005.bf27cdeaf0b8c4647ecd61b1d09da613"
[6] "00006.253ea2f9a9cc36fa0b1129b04b806608"
```

```
head(list.files(path = file.path(data.dir, "spam_2")))
```

```
[1] "00001.317e78fa8ee2f54cd4890fdc09ba8176"
[2] "00002.9438920e9a55591b18e60d1ed37d992b"
[3] "00003.590eff932f8704d8b0fcbe69d023b54d"
[4] "00004.bdcc075fa4beb5157b5dd6cd41d8887b"
[5] "00005.ed0aba4d386c5e62bc737cf3f0ed9589"
[6] "00006.3ca1f399ccda5d897fecb8c57669a283"
```

```
directories <- paste(data.dir, list.files(data.dir), sep = .Platform$file.sep)

file_counts <- sapply(directories, function(dir) length(list.files(dir)))

total_files <- sum(file_counts)
total_files
```

```
[1] 9353
```

```
file_counts
```

```
                D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham
                                                                                5052
              D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham_2
```

```
                                                                                1401
                  D:/Projects/Statistical-Computing/Case Studies/datasets/spam/hard_ham
                                                                                 501
                     D:/Projects/Statistical-Computing/Case Studies/datasets/spam/spam
                                                                                1001
                   D:/Projects/Statistical-Computing/Case Studies/datasets/spam/spam_2
                                                                                1398
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/spamAssassinDerivedDF.rda
                                                                                   0
```

```
idx <- c(1:5, 15, 27, 68, 69, 329, 404, 427, 516, 852, 971)

fn <- list.files(directories[1], full.names = T)[idx]

sampleEmail <- sapply(fn, readLines)
```

Text Mining and Naive Bayes Classification

```
msg <- sampleEmail[[1]]
which(msg == "")[1]
```

```
[1] 63
```

```
match("", msg)
```

```
[1] 63
```

```
splitPoint <- match("", msg)

msg[ (splitPoint - 2):(splitPoint + 6)]
```

```
[1] "List-Archive: <https://listman.spamassassin.taint.org/mailman/private/exmh-workers/>"
[2] "Date: Thu, 22 Aug 2002 18:26:25 +0700"
[3] ""
[4] "    Date:        Wed, 21 Aug 2002 10:54:46 -0500"
[5] "    From:        Chris Garrigues <cwg-dated-1030377287.06fa6d@DeepEddy.Com>"
[6] "    Message-ID:  <1029945287.4797.TMDA@deepeddy.vircio.com>"
[7] ""
[8] ""
[9] "  | I can't reproduce this error."
```

```
header <- msg[1:(splitPoint - 1)]
body <- msg[ -(1:splitPoint) ]
```

```
splitMessage <- function(msg) {
   splitPoint <- match("", msg)
```

```
   header <- msg[ 1:(splitPoint - 1)]
   body <- msg[ -(1:splitPoint)]

   return(list(header = header, body = body))
}

sampleSplit <- lapply(sampleEmail, splitMessage)

header <- sampleSplit[[1]]$header
grep("Content-Type", header)
```

```
[1] 46
```

```
grep("multi", tolower(header))
```

```
integer(0)
```

```
header[46]
```

```
[1] "Content-Type: text/plain; charset=us-ascii"
```

```
headerList <- lapply(sampleSplit, function(msg) msg$header)

CTloc <- sapply(headerList, grep, pattern = "Content-Type")

sapply(headerList, function(header) {
   CTloc <- grep("Content-Type", header)
   if( length(CTloc) == 0) return(NA)
   CTloc
})
```

```
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00001.7c53336b37003a9286a

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00002.9c4069e25e1ef370c07

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00003.860e3c3cee1b42ead71

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00004.864220c5b6930b209cc

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00005.bf27cdeaf0b8c4647ec

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00014.cb20e10b2bfcb8210a1

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00025.d685245bdc4444f44fa

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00062.009f5a1a8fa88f0b382
```

```
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00063.0acbc484a73f0e0b727

 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/0030.77828e31de08ebb58b5

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00368.f86324a03e7ae7070cc

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00389.8606961eaeef7b921ce

 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/0047.5c3e049737a2813d4ac

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00775.0e012f373467846510d

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00883.c44a035e7589e83076b
```

```
hasAttach <- sapply(headerList, function(header) {
   CTloc <- grep("Content-Type", header)

   if(length(CTloc) == 0) return(F)
   grepl("multi", tolower(header[CTloc]))
})

hasAttach
```

```
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00001.7c53336b37003a9286a

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00002.9c4069e25e1ef370c07

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00003.860e3c3cee1b42ead71

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00004.864220c5b6930b209cc

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00005.bf27cdeaf0b8c4647ec

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00014.cb20e10b2bfcb8210a1

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00025.d685245bdc4444f44fa

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00062.009f5a1a8fa88f0b382

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00063.0acbc484a73f0e0b727

 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/0030.77828e31de08ebb58b5

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00368.f86324a03e7ae7070cc
```

```
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00389.8606961eaeef7b921ce

 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/0047.5c3e049737a2813d4ac

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00775.0e012f373467846510d

D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00883.c44a035e7589e83076b
```

```
header <- sampleSplit[[6]]$header
boundaryIdx <- grep("boundary=", header)
header[boundaryIdx]
```

```
[1] "    boundary=\"==_Exmh_-1317289252P\";"
```

```
sub(".*boundary=\"(.*)\";.*", "\\1", header[boundaryIdx])
```

```
[1] "==_Exmh_-1317289252P"
```

```
header2 <- headerList[[9]]
boundaryIdx2 <- grep("boundary=", header2)
header2[boundaryIdx2]
```

```
[1] "Content-Type: multipart/alternative; boundary=Apple-Mail-2-874629474"
```

```
sub('.*boundary="(.*)";.*', "\\1", header2[boundaryIdx2])
```

```
[1] "Content-Type: multipart/alternative; boundary=Apple-Mail-2-874629474"
```

```
boundary2 <- gsub('"', "", header2[boundaryIdx2])

sub(".*boundary= *(.*);?.*", "\\1", boundary2)
```

```
[1] "Apple-Mail-2-874629474"
```

```
boundary <- gsub('"', "", header[boundaryIdx])
sub(".*boundary= *(.*);?.*", "\\1", boundary)
```

```
[1] "==_Exmh_-1317289252P;"
```

```
getBoundary <- function(header) {
   boundaryIdx <- grep("boundary=", header)
   boundary = gsub('"', "", header[boundaryIdx])
   gsub(".*boundary= *([^;]*);?.*", "\\1", boundary)
}
```

```
boundary <- getBoundary(headerList[[15]])
body <- sampleSplit[[15]]$body

bString <- paste("--", boundary, sep = "")
```

```
bStringLocs <- which(bString == body)
bStringLocs
```

```
[1]  2 35
```

```
eString <- paste("--", boundary, "--", sep = "")
eStringLoc <- which(eString == body)
eStringLoc
```

```
[1] 77
```

```
msg <- body[ (bStringLocs[1] + 1) : (bStringLocs[2] - 1)]
tail(msg)
```

```
[1] ">"     ">Yuck" ">  "   ">"     ""      ""
```

```
msg <- c(msg, body[ (eStringLoc + 1) : length(body) ])
tail(msg)
```

```
[1] ">  " ">"   ""    ""    ""    ""
```

Handle Attachments

Extracting Words from the Message Body

```
head(sampleSplit[[1]]$body)
```

```
[1] "    Date:         Wed, 21 Aug 2002 10:54:46 -0500"
[2] "    From:         Chris Garrigues <cwg-dated-1030377287.06fa6d@DeepEddy.Com>"
[3] "    Message-ID:   <1029945287.4797.TMDA@deepeddy.vircio.com>"
[4] ""
[5] ""
[6] "  | I can't reproduce this error."
```

```
msg <- sampleSplit[[3]]$body
head(msg)
```

```
[1] "Man Threatens Explosion In Moscow "
[2] ""
[3] "Thursday August 22, 2002 1:40 PM"
[4] "MOSCOW (AP) - Security officers on Thursday seized an unidentified man who"
[5] "said he was armed with explosives and threatened to blow up his truck in"
[6] "front of Russia's Federal Security Services headquarters in Moscow, NTV"
```

Stemming

```
exclude_word_list <- stopwords(kind = "en")
```

Convert To Wordlist

```
tolower(gsub("[[:punct:]0-9[:blank:]]+", " ", msg))
```

```
 [1] "man threatens explosion in moscow "
 [2] ""
 [3] "thursday august pm"
 [4] "moscow ap security officers on thursday seized an unidentified man who"
 [5] "said he was armed with explosives and threatened to blow up his truck in"
 [6] "front of russia s federal security services headquarters in moscow ntv"
 [7] "television reported "
 [8] "the officers seized an automatic rifle the man was carrying then the man"
 [9] "got out of the truck and was taken into custody ntv said no other details"
[10] "were immediately available "
[11] "the man had demanded talks with high government officials the interfax and"
[12] "itar tass news agencies said ekho moskvy radio reported that he wanted to"
[13] "talk with russian president vladimir putin "
[14] "police and security forces rushed to the security service building within"
[15] "blocks of the kremlin red square and the bolshoi ballet and surrounded the"
[16] "man who claimed to have one and a half tons of explosives the news"
[17] "agencies said negotiations continued for about one and a half hours outside"
[18] "the building itar tass and interfax reported citing witnesses "
[19] "the man later drove away from the building under police escort and drove"
[20] "to a street near moscow s olympic penta hotel where authorities held"
[21] "further negotiations with him the moscow police press service said the"
[22] "move appeared to be an attempt by security services to get him to a more"
[23] "secure location "
[24] ""
[25] " yahoo groups sponsor "
[26] " dvds free s p join now"
[27] "http us click yahoo com pt ybb nxieaa mg haa gsolb tm"
[28] " "
[29] ""
[30] "to unsubscribe from this group send an email to "
[31] "forteana unsubscribe egroups com"
[32] ""
[33] " "
[34] ""
[35] "your use of yahoo groups is subject to http docs yahoo com info terms "
[36] ""
```

```
[37] ""
[38] ""
```

```
msg[ c(1, 3, 26, 27) ]
```

```
[1] "Man Threatens Explosion In Moscow "
[2] "Thursday August 22, 2002 1:40 PM"
[3] "4 DVDs Free +s&p Join Now"
[4] "http://us.click.yahoo.com/pt6YBB/NXiEAA/mG3HAA/7gSolB/TM"
```

```
cleanMsg <- tolower(gsub("[[:punct:]0-9[:blank:]]+", " ", msg))
cleanMsg[ c(1, 3, 26, 27) ]
```

```
[1] "man threatens explosion in moscow "
[2] "thursday august pm"
[3] " dvds free s p join now"
[4] "http us click yahoo com pt ybb nxieaa mg haa gsolb tm"
```

```
words <- unlist(strsplit(cleanMsg, "[[:blank:]]+"))

words <- words[ nchar(words) > 1 ]

words <- words[ ! (words %in% exclude_word_list) ]

head(words)
```

```
[1] "man"       "threatens" "explosion" "moscow"    "thursday"  "august"
```

```
findMsgWords <- function(msg, exclude) {

   cleanMsg <- tolower(gsub("[[:punct:]0-9[:blank:]]+", " ", msg))

   words <- unlist(strsplit(cleanMsg, "[[:blank:]]+"))

   keep <- sapply(words, function(word) return(!(word %in% exclude)))


   return(words[ keep ])
}
```

Prep Wrap-Up

```
dropAttach <- function(body, boundary) {

   if(is.null(body)) {
      return("")
   }
```

```
   bString <- paste("--", boundary, sep = "")
   bStringLocs <- which(bString == body)

   eString <- paste("--", boundary, "--", sep = "")
   eStringLoc <- which(eString == body)

   if(length(bStringLocs) == 2) {
      msg <- body[ (bStringLocs[1] + 1) : (bStringLocs[2] - 1)]
   }

   if(length(eStringLoc) > 0) {
      msg <- c(msg, body[ (eStringLoc + 1) : length(body) ])
   }

   return(msg)
}

processAllWords <- function(dirName, stopWords) {
   # read all files in the directory
   fileNames <- list.files(dirName, full.names = T)

   # drop files that are not email, i.e., cmds
   notEmail <- grep("cmds$", fileNames)

   if( length(notEmail) > 0) fileNames <- fileNames[ -notEmail ]

   messages <- lapply(fileNames, readLines, encoding = "latin1")

   # split header and body
   emailSplit <- lapply(messages, splitMessage)

   # put body and header in own lists
   bodyList <- lapply(emailSplit, function(msg) msg$body)
   headerList <- lapply(emailSplit, function(msg) msg$header)
   rm(emailSplit)

   # determine which messages have attachments
   hasAttach <- sapply(headerList, function(header) {

      CTloc <- grep("Content-Type", header)

      if( length(CTloc) == 0) return(0)

      multi <- grep("multi", tolower(header[CTloc]))
```

```
      if( length(multi) == 0 ) return(0)

      multi
   })

   hasAttach <- which(hasAttach > 0)

   # find boundary string for messages with attachments
   boundaries <- sapply(headerList[hasAttach], getBoundary)

   # drop attachments from message body
   bodyList[hasAttach] <- mapply(dropAttach, bodyList[hasAttach],
                                 boundaries, SIMPLIFY = F)

   # extract words from body
   msgWordsList <- lapply(bodyList, findMsgWords, stopWords)

   invisible(msgWordsList)
}
```

Build Email Database

```
msgWordList <- lapply(directories, processAllWords, stopWords = exclude_word_list)
```

```
Warning in FUN(X[[i]], ...): incomplete final line found on 'D:/
Projects/Statistical-Computing/Case Studies/datasets/spam/hard_ham/
00228.0eaef7857bbbf3ebf5edbbdae2b30493'

Warning in FUN(X[[i]], ...): incomplete final line found on 'D:/
Projects/Statistical-Computing/Case Studies/datasets/spam/hard_ham/
0231.7c6cc716ce3f3bfad7130dd3c8d7b072'

Warning in FUN(X[[i]], ...): incomplete final line found on 'D:/
Projects/Statistical-Computing/Case Studies/datasets/spam/hard_ham/
0250.7c6cc716ce3f3bfad7130dd3c8d7b072'

Warning in FUN(X[[i]], ...): incomplete final line found on 'D:/
Projects/Statistical-Computing/Case Studies/datasets/spam/spam/
00136.faa39d8e816c70f23b4bb8758d8a74f0'

Warning in FUN(X[[i]], ...): incomplete final line found on 'D:/
Projects/Statistical-Computing/Case Studies/datasets/spam/spam/
0143.260a940290dcb61f9327b224a368d4af'
```

```
numMsgs <- sapply(msgWordList, length)
numMsgs
```

```
[1] 5051 1400  500 1000 1397    0
```

```
isSpam <- c(rep(FALSE, numMsgs[1]),
            rep(FALSE, numMsgs[2]),
            rep(FALSE, numMsgs[3]),
            rep(TRUE, numMsgs[4]),
            rep(TRUE, numMsgs[5]))

msgWordsList <- unlist(msgWordList, recursive = F)
```

Naive Bayes Classifier Implementation

Train / Test Split

```
numEmail <- length(isSpam)

numSpam <- sum(isSpam)
numHam <- numEmail - numSpam

set.seed(418910)

testSpamIdx <- sample(numSpam, size = floor(numSpam/3))
testHamIdx <- sample(numHam, size = floor(numHam/3))

testMsgWords <- c((msgWordsList[isSpam])[testSpamIdx],
                  (msgWordsList[!isSpam])[testHamIdx])

trainMsgWords <- c((msgWordsList[isSpam])[ - testSpamIdx ],
                   (msgWordsList[!isSpam])[ - testHamIdx])

testIsSpam <- rep(c(T, F),
                  c(length(testSpamIdx), length(testHamIdx)))

trainIsSpam <- rep(c(T, F),
                   c(numSpam - length(testSpamIdx),
                     numHam - length(testHamIdx)))
```

Probability Estimates from Training Sample

```
bow <- unique(unlist(trainMsgWords))

length(bow)
```

```
[1] 69502
```

```
spamWordCounts <- rep(0, length(bow))

names(spamWordCounts) = bow

tmp <- lapply(trainMsgWords[trainIsSpam], unique)
tt <- table( unlist(tmp) )
spamWordCounts[ names(tt) ] = tt

spamWordsProbs <- (spamWordCounts + 0.5) / (sum(trainIsSpam) + 0.5)

spamWordsProbs[1:20]
```

```
                fight         risk       cancer         http          www
0.0003127932 0.0109477635 0.0910228339 0.0165780419 0.8686268377 0.4876446669
     adclick           ws            p          cfm            o            s
0.0147012825 0.0240850798 0.4644979668 0.0165780419 0.1316859556 0.5595871129
          pk         slim   guaranteed         lose          lbs         days
0.0159524554 0.0140756960 0.1129183610 0.0672505474 0.0153268689 0.1467000313
         get        child
0.4388489209 0.0184548014
```

```
hamWordCounts <- rep(0, length(bow))

names(hamWordCounts) = bow

tmp <- lapply(trainMsgWords[ - trainIsSpam], unique)
tt <- table( unlist(tmp) )
hamWordCounts[ names(tt) ] = tt

hamWordsProbs <- (hamWordCounts + 0.5) / (sum(!trainIsSpam) + 0.5)

probs <- log(spamWordsProbs) - log(hamWordsProbs)

head(probs)
```

```
               fight       risk     cancer       http        www
 1.0644626 -0.2553866  0.6999150  0.4600436 -0.2252153 -0.4263420
```

```
wordsList <- trainMsgWords
spam <- trainIsSpam

make_words_valid_columns <- function( words, all_words ) {

   word_counts <- rep(0, length(all_words))
   names(word_counts) <- all_words

   tmp <- lapply(words, unique)
   tt <- table( unlist(tmp) )
   word_counts[ names(tt) ] = tt

   return(word_counts)
}

computeFreqs <- function(wordsList, spam, bow = unique(unlist(wordsList))) {

   all_words <- unique(bow)

   # create a matrix for spam, ham, and log odds
   wordTable <- matrix(0.5, nrow = 2, ncol = length(bow))
   colnames(wordTable) <- all_words
   rownames(wordTable) <- c( "presentLogOdds",
                             "absentLogOdds")

   # for each spam message, add 1 to the counts for words in messsage

   spam_all <- wordsList[spam]
   spam_words <- make_words_valid_columns( spam_all, all_words )

   wordTable <- rbind(wordTable, spam_words + 0.5)
   rownames(wordTable)[3] <- "spam"

   # Similarly for ham messages

   ham_all <- wordsList[ !spam ]

   ham_words <- make_words_valid_columns( ham_all, all_words )

   wordTable <- rbind(wordTable, ham_words + 0.5)
   rownames(wordTable)[4] <- "ham"

   head(wordTable[, 1:20])
```

```
   # find the total number of spam and ham
   numSpam <- sum(spam)
   numHam <- length(spam) - numSpam

   # prob (word|spam) and prob(words|ham)
   wordTable["spam", ] <- wordTable["spam", ] / (numSpam + 0.5)
   wordTable["ham", ] <- wordTable["ham", ] / (numHam + 0.5)

   head(wordTable[, 1:20])

   # log odds
   wordTable["presentLogOdds", ] =
      log(wordTable["spam", ]) - log(wordTable["ham", ])

   wordTable["absentLogOdds", ] =
      log((1 - wordTable["spam", ])) - log((1 - wordTable["ham", ]))

   invisible(wordTable)
}
```

```
trainTable <- computeFreqs(trainMsgWords, trainIsSpam)
```

```
Warning in rbind(wordTable, spam_words + 0.5): number of columns of result is
not a multiple of vector length (arg 2)
```

```
Warning in rbind(wordTable, ham_words + 0.5): number of columns of result is not
a multiple of vector length (arg 2)
```

```
# peek the prob table
head(trainTable[, 1:10])
```

```
                                fight        risk       cancer        http
presentLogOdds  1.0644626288  0.0246908402  1.86258857  1.184606941  0.09645377
absentLogOdds  -0.0002049499 -0.0002699187 -0.08120135 -0.011633430 -0.47496148
spam            0.0003127932  0.0109477635  0.09102283  0.016578042  0.86862684
ham             0.0001078865  0.0106807638  0.01413313  0.005070666  0.78875823
                      www       adclick          ws         p          cfm
presentLogOdds -0.1717619  4.9146102305  0.68088023  1.347442  0.114773617
absentLogOdds   0.1964494 -0.0147025249 -0.01211377 -0.495893 -0.001826225
spam            0.4876447  0.0147012825  0.02408508  0.464498  0.016578042
ham             0.5790269  0.0001078865  0.01219117  0.120725  0.014780451
```

Classifying New Messages

```
newMsg <- testMsgWords[[1]]

# only look at words we have classified
newMsg <- newMsg[ !is.na(match(newMsg, colnames(trainTable)))]

present <- colnames(trainTable) %in% newMsg

sum( trainTable["presentLogOdds", present]) +
   sum( trainTable["absentLogOdds", !present])
```

```
[1] 29.76454
```

```
newMsg <- testMsgWords[[ which(!testIsSpam)[ 1 ] ]]
newMsg <- newMsg[ !is.na(match(newMsg, colnames(trainTable)))]
present <- (colnames(trainTable) %in% newMsg)

sum(trainTable["presentLogOdds", present]) +
   sum(trainTable["absentLogOdds", !present])
```

```
[1] -151.9407
```

```
computeMsgLLR <- function(words, freqTable) {

   # discard words not in training data
   words <- words[!is.na(match(words, colnames(freqTable)))]

   # Find which words are present
   present <- colnames(freqTable) %in% words

   sum(freqTable["presentLogOdds", present]) +
      sum(freqTable["absentLogOdds", !present])
}
```

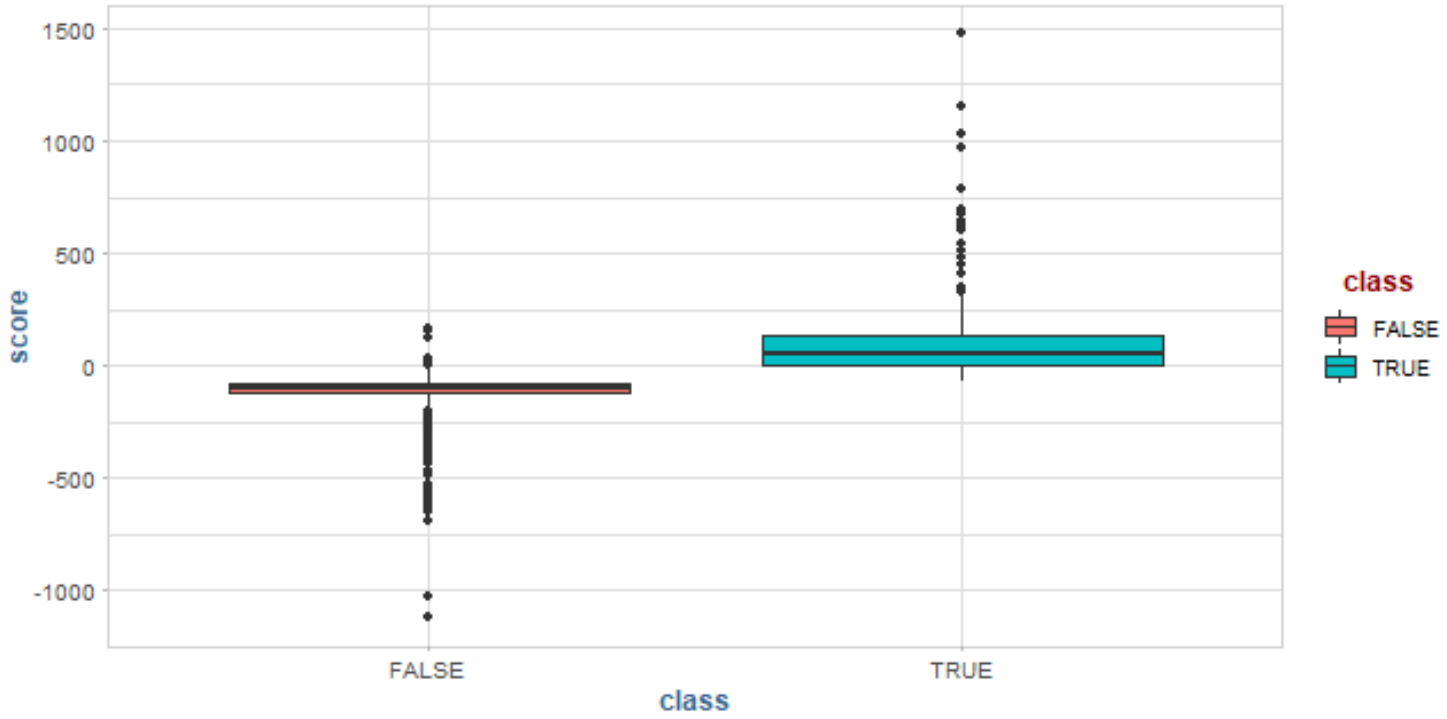
```
testLLR <- sapply(testMsgWords, computeMsgLLR, trainTable)
```

```
tapply(testLLR, testIsSpam, summary)
```

```
$`FALSE`
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1117.24  -125.38   -95.56  -113.54   -76.09   162.06

$`TRUE`
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 -66.359    6.614   52.117   85.706  129.858 1473.652
```

```
results_df <- data.table( score = testLLR, class = testIsSpam )
```

```
ggplot(results_df, aes(score, class, fill = class)) +
   geom_boxplot() +
   coord_flip()
```



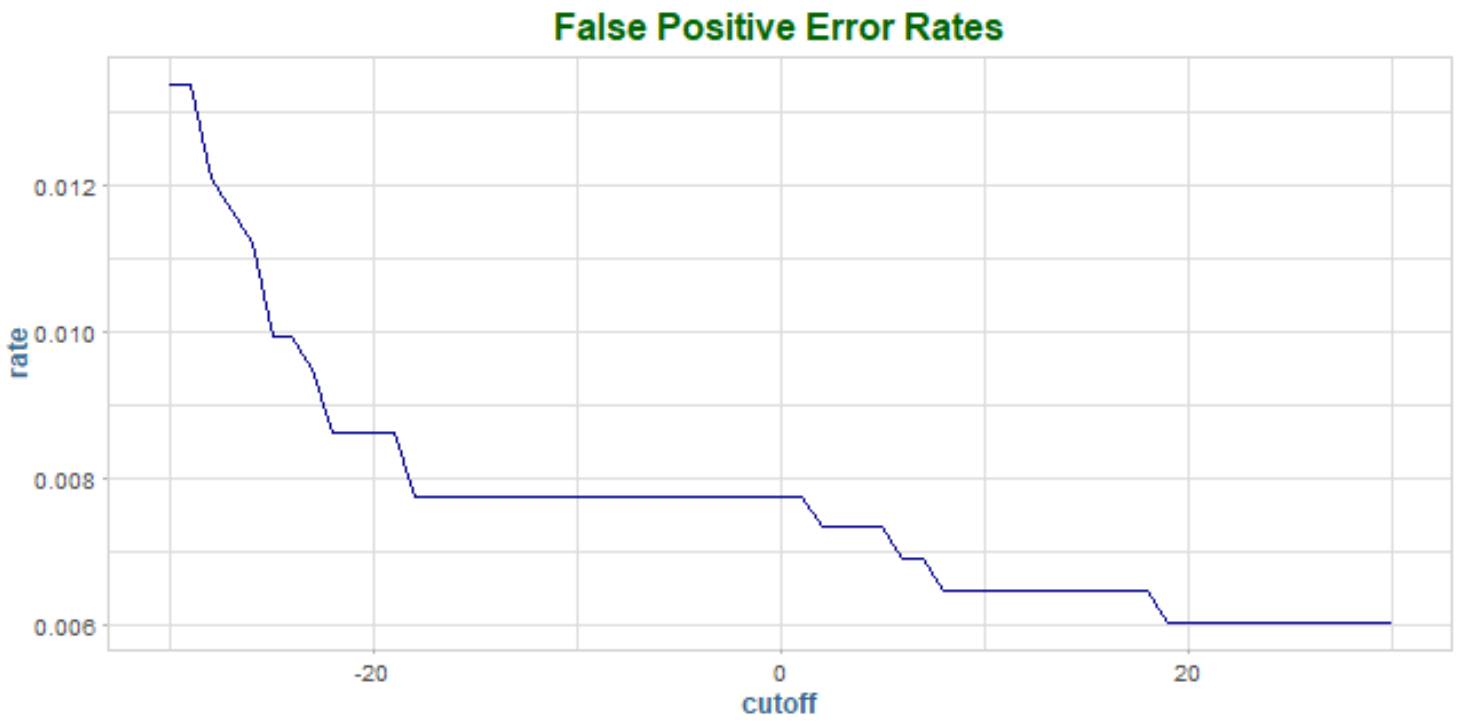

```
typeIErrorRate <- function(tau, llrVals, spam) {
   classify <- llrVals > tau
   sum(classify & !spam) / sum(!spam)
}

typeIErrorRate(0, testLLR, testIsSpam)
```

```
[1] 0.007768666
```

```
typeIErrorRate(-20, testLLR, testIsSpam)
```

```
[1] 0.008631852
```

```
error_rates <- sapply(seq(-30, 30, 1), function(cutoff) c(cutoff = cutoff, rate = typeIErrorRa

er_df <- data.table(t(error_rates))

ggplot(er_df, aes(cutoff, rate)) +
   geom_line(col = "darkblue") +
   labs(title = "False Positive Error Rates")
```




```
typeIErrorRates <- function(llrVals, isSpam) {
   o <- order(llrVals)
   llrVals <- llrVals[o]
   isSpam <- isSpam[o]

   idx <- which(!isSpam)
   N <- length(idx)
   list(error = (N:1)/N, values = llrVals[idx])
}
```

Computational Considerations

```
smallNums <- rep((1/2)^40, 2000000)
largeNum <- 10000

print(sum(smallNums), digits = 20)
```

```
[1] 1.8189894035458565e-06
```

```
print(largeNum + sum(smallNums), digits = 20)
```

```
[1] 10000.000001818989
```

```
for(i in 1:length(smallNums)) {
   largeNum <- largeNum + smallNums[i]
```

```
}

print(largeNum, digits = 20)
```

```
[1] 10000
```

Recursive Partitioning and Classification Trees

Revised E-mail Data Structure

```
header <- sampleSplit[[1]]$header

header[1:12]
```

```
 [1] "From exmh-workers-admin@redhat.com  Thu Aug 22 12:36:23 2002"
 [2] "Return-Path: <exmh-workers-admin@spamassassin.taint.org>"
 [3] "Delivered-To: zzzz@localhost.netnoteinc.com"
 [4] "Received: from localhost (localhost [127.0.0.1])"
 [5] "\tby phobos.labs.netnoteinc.com (Postfix) with ESMTP id D03E543C36"
 [6] "\tfor <zzzz@localhost>; Thu, 22 Aug 2002 07:36:16 -0400 (EDT)"
 [7] "Received: from phobos [127.0.0.1]"
 [8] "\tby localhost with IMAP (fetchmail-5.9.0)"
 [9] "\tfor zzzz@localhost (single-drop); Thu, 22 Aug 2002 12:36:16 +0100 (IST)"
[10] "Received: from listman.spamassassin.taint.org (listman.spamassassin.taint.org [66.187.233
[11] "    dogma.slashnull.org (8.11.6/8.11.6) with ESMTP id g7MBYrZ04811 for"
[12] "    <zzzz-exmh@spamassassin.taint.org>; Thu, 22 Aug 2002 12:34:53 +0100"
```

```
header[1] = sub("^From", "Top-From:", header[1])

headerPieces <- read.dcf(textConnection(header), all = T)

headerPieces[, "Delivered-To"]
```

```
[[1]]
[1] "zzzz@localhost.netnoteinc.com"
[2] "exmh-workers@listman.spamassassin.taint.org"
```

```
headerVec <- unlist(headerPieces)
dupKeys <- sapply(headerPieces, function(x) length(unlist(x)))
names(headerVec) <- rep(colnames(headerPieces), dupKeys)

headerVec[ which(names(headerVec) == "Delivered-To") ]
```

```
                      Delivered-To
"zzzz@localhost.netnoteinc.com"
```

```
                                Delivered-To
"exmh-workers@listman.spamassassin.taint.org"
```

```
length(headerVec)
```

```
[1] 36
```

```
length(unique(names(headerVec)))
```

```
[1] 26
```

```
processHeader <- function(header) {
   # modify the first line to create a key:value pair
   header[1] <- sub("^From", "Top-From:", header[1])

   headerMat <- read.dcf(textConnection(header), all = T)
   headerVec <- unlist(headerMat)

   dupKeys <- sapply(headerMat, function(x) length(unlist(x)))
   names(headerVec) <- rep(colnames(headerMat), dupKeys)

   return(headerVec)
}

headerList <- lapply(sampleSplit,
                     function(msg) {
                        processHeader(msg$header)
                     })

contentTypes <- sapply(headerList, function(header)
   header["Content-Type"])

names(contentTypes) <- NULL

contentTypes
```

```
 [1] "text/plain; charset=us-ascii"
 [2] "text/plain; charset=US-ASCII"
 [3] "text/plain; charset=US-ASCII"
 [4] "text/plain; charset=\"us-ascii\""
 [5] "text/plain; charset=US-ASCII"
 [6] "multipart/signed;\nboundary=\"==_Exmh_-1317289252P\";\nmicalg=pgp-sha1;\nprotocol=\"appli
 [7] NA
 [8] "multipart/alternative;\nboundary=\"----=_NextPart_000_00C1_01C25017.F2F04E20\""
 [9] "multipart/alternative; boundary=Apple-Mail-2-874629474"
[10] "multipart/signed;\nboundary=\"==_Exmh_-518574644P\";\nmicalg=pgp-sha1;\nprotocol=\"applic
[11] "multipart/related;\nboundary=\"------------090602010909000705010009\""
```

```
[12] "multipart/signed;\nboundary=\"==_Exmh_-451422450P\";\nmicalg=pgp-sha1;\nprotocol=\"applic
[13] "multipart/signed;\nboundary=\"==_Exmh_267413022P\";\nmicalg=pgp-sha1;\nprotocol=\"applica
[14] "multipart/mixed;\nboundary=\"----=_NextPart_000_0005_01C26412.7545C1D0\""
[15] "multipart/alternative;\nboundary=\"------------080209060700030309080805\""
```

Attachments Revisited

```
hasAttach <- grep("^ *multi", tolower(contentTypes))
hasAttach
```

```
[1]  6  8  9 10 11 12 13 14 15
```

```
boundaries <- getBoundary(contentTypes[ hasAttach ])
boundaries
```

```
[1] "==_Exmh_-1317289252P"
[2] "----=_NextPart_000_00C1_01C25017.F2F04E20"
[3] "Apple-Mail-2-874629474"
[4] "==_Exmh_-518574644P"
[5] "------------090602010909000705010009"
[6] "==_Exmh_-451422450P"
[7] "==_Exmh_267413022P"
[8] "----=_NextPart_000_0005_01C26412.7545C1D0"
[9] "------------080209060700030309080805"
```

```
boundary <- boundaries[9]
body <- sampleSplit[[15]]$body

bString <- paste("--", boundary, sep = "")
bStringLocs <- which(bString == body)
bStringLocs
```

```
[1]  2 35
```

```
eString <- paste("--", boundary, "--", sep = "")
eStringLoc <- which(eString == body)
eStringLoc
```

```
[1] 77
```

```
range <- diff(c(bStringLocs[-1], eStringLoc))

body[1:range]
```

```
 [1] ""
 [2] "--------------080209060700030309080805"
 [3] "Content-Type: text/plain; charset=US-ASCII; format=flowed"
 [4] "Content-Transfer-Encoding: 7bit"
```

```
 [5] ""
 [6] "I actually thought of this kind of active chat at AOL (in 1996 I think), "
 [7] "bringing up ads based on what was being discussed and other features. "
 [8] "For a while, the VP of dev. (now still CTO I think) was really hot on "
 [9] "the idea and they discussed patenting it. Then they lost interest. "
[10] "Probably a good thing."
[11] ""
[12] "sdw"
[13] ""
[14] "Lorin Rivers wrote:"
[15] ""
[16] ">On 10/2/02 12:00 PM, \"Mr. FoRK\" <fork_list@hotmail.com> wrote:"
[17] ">  "
[18] ">"
[19] ">>What about a situation where you don't directly ask/talk to the bot, but"
[20] ">>they listen in and advise/correct/interject/etc?"
[21] ">>example: two people discussing trips, etc. may trigger a weather bot to"
[22] ">>mention what the forecast says - without directly being asked."
[23] ">>    "
[24] ">>"
[25] ">"
[26] ">My guess is it's more insidious than that, it's going to be ActiveSpam."
[27] ">"
[28] ">\"Oh, you're going to Seattle? I can get you airline tickets for less\""
[29] ">"
[30] ">Yuck"
[31] ">  "
[32] ">"
[33] ""
[34] ""
[35] "--------------080209060700030309080805"
[36] "Content-Type: text/html; charset=US-ASCII"
[37] "Content-Transfer-Encoding: 7bit"
[38] ""
[39] "<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">"
[40] "<html>"
[41] "<head>"
[42] "  <title></title>"
```

```
processAttach <- function(body, contentType ) {

   boundary <- getBoundary(contentType)

   bString <- paste("--", boundary, sep = "")
   bStringLocs <- which(bString == body)
```

```
   eString <- paste("--", boundary, "--", sep = "")
   eStringLoc <- which(eString == body)

   n <- length(body)

   if(length(bStringLocs) == 2) {

      bodyContent <- body[(bStringLocs[1] + 2):(bStringLocs[2] - 1)]

      emptyLines <- which(bodyContent == "")
      bodyContent <- bodyContent[ - emptyLines]

      attachContent <- body[(bStringLocs[2] + 1):n]

      aLen <- diff(c(bStringLocs[-1], eStringLoc))
      aType <- body[bStringLocs[-1] + 1]

      if(length(aLen) == length(aType)) {
         attachments <- data.frame(aLen = aLen, aType = aType)
      } else {
         attachments <- data.frame(aLen = c(), aType = c())
      }

   } else {
      if( length(bStringLocs) == 0 ) {
         bodyContent <- body
      } else {
         bodyContent = body
      }
      attachments <- data.frame(aLen = c(), aType = c())
   }

   return(list(body = bodyContent, attachDF = attachments ))
}
```

More E-Mails

```
bodyList <- lapply(sampleSplit, function(msg) msg$body)
attList <- mapply(processAttach, bodyList[hasAttach],
                  contentTypes[hasAttach], SIMPLIFY = F)

lens <- sapply(attList, function(processedA)
                           processedA$attachDF$aLen)
```

```
readEmail <- function(dirName) {
   # retrieve the names of files in the directory
   fileNames <- list.files(dirName, full.names = T)

   # drop files that are not email
   notEmail <- grep("cmds$", fileNames)

   if( length(notEmail) > 0 ) fileNames = fileNames[ - notEmail ]

   # read all files in the directory
   lapply(fileNames, readLines, encoding = "latin1")
}

processAllEmail <- function(dirName, isSpam = F) {

   # read all files in the directory
   messages <- readEmail(dirName)

   fileNames <- names(messages)
   n <- length(messages)

   # split header from body
   eSplit <- lapply(messages, splitMessage)
   rm(messages)

   # process header as named character vector
   headerList <- lapply(eSplit, function(msg)
                           processHeader(msg$header))

   # extractd content-type key
   contentTypes <- sapply(headerList, function(header)
                                       header["Content-Type"])

   # extract the body
   bodyList <- lapply(eSplit, function(msg) msg$body)
   rm(eSplit)

   # which email have attachements
   hasAttach <- grep("^ *multi", tolower(contentTypes))

   # get summary stats for attachments and the shorter body
   attList <- mapply(processAttach, bodyList[hasAttach],
                     contentTypes[hasAttach], SIMPLIFY = F)
```

```
   bodyList[hasAttach] <- lapply(attList, function(attEl)
                                          attEl$body)

   attachInfo <- vector("list", length = n)
   attachInfo[ hasAttach ] <- lapply(attList,
                                     function(attEl) attEl$attachDf)

   # prepare return structure
   emailList <- mapply(function(header, body, attach, isSpam) {
      list(isSpam = isSpam, header = header,
           body = body, attach = attach)
   },
   headerList, bodyList, attachInfo,
   rep(isSpam, n), SIMPLIFY = F)

   names(emailList) <- fileNames

   invisible(emailList)
}
```

```
emailStruct <- mapply(processAllEmail, directories,
                      isSpam = rep( c(F, T), 3:2))
```

```
Warning in FUN(X[[i]], ...): incomplete final line found on 'D:/
Projects/Statistical-Computing/Case Studies/datasets/spam/hard_ham/
00228.0eaef7857bbbf3ebf5edbbdae2b30493'

Warning in FUN(X[[i]], ...): incomplete final line found on 'D:/
Projects/Statistical-Computing/Case Studies/datasets/spam/hard_ham/
0231.7c6cc716ce3f3bfad7130dd3c8d7b072'

Warning in FUN(X[[i]], ...): incomplete final line found on 'D:/
Projects/Statistical-Computing/Case Studies/datasets/spam/hard_ham/
0250.7c6cc716ce3f3bfad7130dd3c8d7b072'

Warning in FUN(X[[i]], ...): incomplete final line found on 'D:/
Projects/Statistical-Computing/Case Studies/datasets/spam/spam/
00136.faa39d8e816c70f23b4bb8758d8a74f0'

Warning in FUN(X[[i]], ...): incomplete final line found on 'D:/
Projects/Statistical-Computing/Case Studies/datasets/spam/spam/
0143.260a940290dcb61f9327b224a368d4af'

Warning in mapply(processAllEmail, directories, isSpam = rep(c(F, T), 3:2)):
longer argument not a multiple of length of shorter
```

```
emailStruct <- unlist(emailStruct, recursive = F)
```

```
sampleStruct <- emailStruct[ 1:15 ]
```

Deriving Variables from the email Messages

```
header <- sampleStruct[[1]]$header
subject <- header["Subject"]
els <- strsplit(subject, "")
all(els %in% LETTERS)
```

```
[1] FALSE
```

```
testSubject <- c("DEAR MADAM", "WINNER!", "")

els <- strsplit(testSubject, "")
sapply(els, function(subject) all(subject %in% LETTERS))
```

```
[1] FALSE FALSE  TRUE
```

```
gsub("[[:punct:] ]", "", testSubject)
```

```
[1] "DEARMADAM" "WINNER"    ""
```

```
gsub("[^[:alpha:]]", "", testSubject)
```

```
[1] "DEARMADAM" "WINNER"    ""
```

```
isYelling <- function(msg) {
   if( "Subject" %in% names(msg$header) ) {
      el <- gsub("[^[:alpha:]]", "", msg$header["Subject"])

      if ( nchar(el) > 0 )
         nchar(gsub("[A-Z]", "", el) < 1 )
      else
         FALSE
   } else {
      NA
   }
}

perCaps <- function(msg) {

   body <- paste(msg$body, collapse = "")

   # Return NA if the body of the message is "empty"
   if(length(body) == 0 || nchar(body) == 0) return (NA)
```

```
   # Eliminate non-alpha characters
   body <- gsub("[^[:alpha:]]", "", body)
   capText <- gsub("[^A-Z]", "", body)
   100 * nchar(capText)/nchar(body)
}
```

```
sapply(sampleStruct, perCaps)
```

```
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham1
                                                                4.451039
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham2
                                                                7.491289
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham3
                                                                7.436096
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham4
                                                                5.090909
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham5
                                                                6.116643
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham6
                                                                7.625272
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham7
                                                                6.343714
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham8
                                                                6.617647
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham9
                                                                3.161361
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham10
                                                                4.451039
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham11
                                                                5.564648
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham12
                                                                4.785894
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham13
                                                                4.454023
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham14
                                                                3.488372
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham15
                                                                8.275862
```

```
funcList <- list(

   isRe = function(msg) {
      "Subject" %in% names(msg$header) &&
         length(grep("^[ ]*Re:", msg$header[["Subject"]])) > 0
   },
```

```
   numLines = function(msg) {
      length(msg$body)
   },
   isYelling = function(msg) {
      if( "Subject" %in% names(msg$header) ) {
         el <- gsub("[^[:alpha:]]", "", msg$header["Subject"])

         if ( nchar(el) > 0 )
            nchar(gsub("[A-Z]", "", el) < 1 )
         else
            FALSE
      } else {
         NA
      }
   },
   perCaps = function(msg) {

      body <- paste(msg$body, collapse = "")

      # Return NA if the body of the message is "empty"
      if(length(body) == 0 || nchar(body) == 0) return (NA)

      # Eliminate non-alpha characters
      body <- gsub("[^[:alpha:]]", "", body)
      capText <- gsub("[^A-Z]", "", body)
      100 * nchar(capText)/nchar(body)
   }
)
```

```
lapply(funcList, function(func)
   sapply(sampleStruct, function(msg) func(msg)))
```

```
$isRe
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham1
                                                                   TRUE
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham2
                                                                  FALSE
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham3
                                                                  FALSE
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham4
                                                                  FALSE
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham5
                                                                   TRUE
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham6
                                                                   TRUE
```

```
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham7
                                                                  FALSE
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham8
                                                                   TRUE
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham9
                                                                  FALSE
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham10
                                                                   TRUE
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham11
                                                                  FALSE
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham12
                                                                  FALSE
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham13
                                                                   TRUE
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham14
                                                                  FALSE
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham15
                                                                   TRUE

$numLines
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham1
                                                                     50
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham2
                                                                     26
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham3
                                                                     38
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham4
                                                                     32
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham5
                                                                     31
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham6
                                                                     25
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham7
                                                                     38
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham8
                                                                     39
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham9
                                                                    126
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham10
                                                                     50
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham11
                                                                     19
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham12
                                                                     20
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham13
```

```
                                                                    27
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham14
                                                                    28
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham15
                                                                    35

$isYelling
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham1.Subject
                                                                     5
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham2.Subject
                                                                     5
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham3.Subject
                                                                     5
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham4.Subject
                                                                     5
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham5.Subject
                                                                     5
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham6.Subject
                                                                     5
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham7.Subject
                                                                     5
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham8.Subject
                                                                     5
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham9.Subject
                                                                     5
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham10.Subject
                                                                     5
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham11.Subject
                                                                     5
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham12.Subject
                                                                     5
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham13.Subject
                                                                     5
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham14.Subject
                                                                     5
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham15.Subject
                                                                     5

$perCaps
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham1
                                                              4.451039
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham2
                                                              7.491289
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham3
                                                              7.436096
```

```
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham4
                                                                5.090909
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham5
                                                                6.116643
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham6
                                                                7.625272
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham7
                                                                6.343714
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham8
                                                                6.617647
 D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham9
                                                                3.161361
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham10
                                                                4.451039
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham11
                                                                5.564648
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham12
                                                                4.785894
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham13
                                                                4.454023
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham14
                                                                3.488372
D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham15
                                                                8.275862
```

```
createDerivedF <- function(email = emailStruct, operations = funcList,
                           verbose = F)
{
   els <- lapply(names(operations),
                 function(id) {
                     if(verbose) print(id)
                     e <- operations[[id]]
                     v <- if(is.function(e))
                           sapply(email, e)
                        else
                           sapply(email, function(msg) eval(e))
                     v
                  })

   df <- as.data.frame(els)
   names(df) <- names(operations)

   invisible(df)
}
```

```
sampleDF <- createDerivedF(sampleStruct)
```

```
spam_data <- file.path(data.dir, "spamAssassinDerivedDF.rda")

load(spam_data)
```

```
perCaps2 <- function(msg) {

   body <- paste(msg$body, collapse = "")

   # return NA if the body of the message is "empty"
   if(length(body) == 0 || nchar(body) == 0) return(NA)

   # eliminate non-alpha characters and empty lines
   body <- gsub("[^[:alpha:]]", "", body)
   els <- unlist(strsplit(body, ""))
   ctCap <- sum(els %in% LETTERS)
   100 * ctCap / length(els)
}
```

```
pC <- sapply(emailStruct, perCaps)
pC2 <- sapply(emailStruct, perCaps2)

identical(pC, pC2)
```

```
[1] TRUE
```

```
indNA <- which(is.na(emailDF$subExcCt))

indNoSubject <- which(sapply(emailStruct,
                             function(msg)
                                !("Subject" %in% names(msg$header))))

all(indNA == indNoSubject)
```

```
Warning in indNA == indNoSubject: longer object length is not a multiple of
shorter object length
```

```
[1] FALSE
```

```
all(emailDF$bodyCharCt > emailDF$numLines)
```

```
[1] FALSE
```

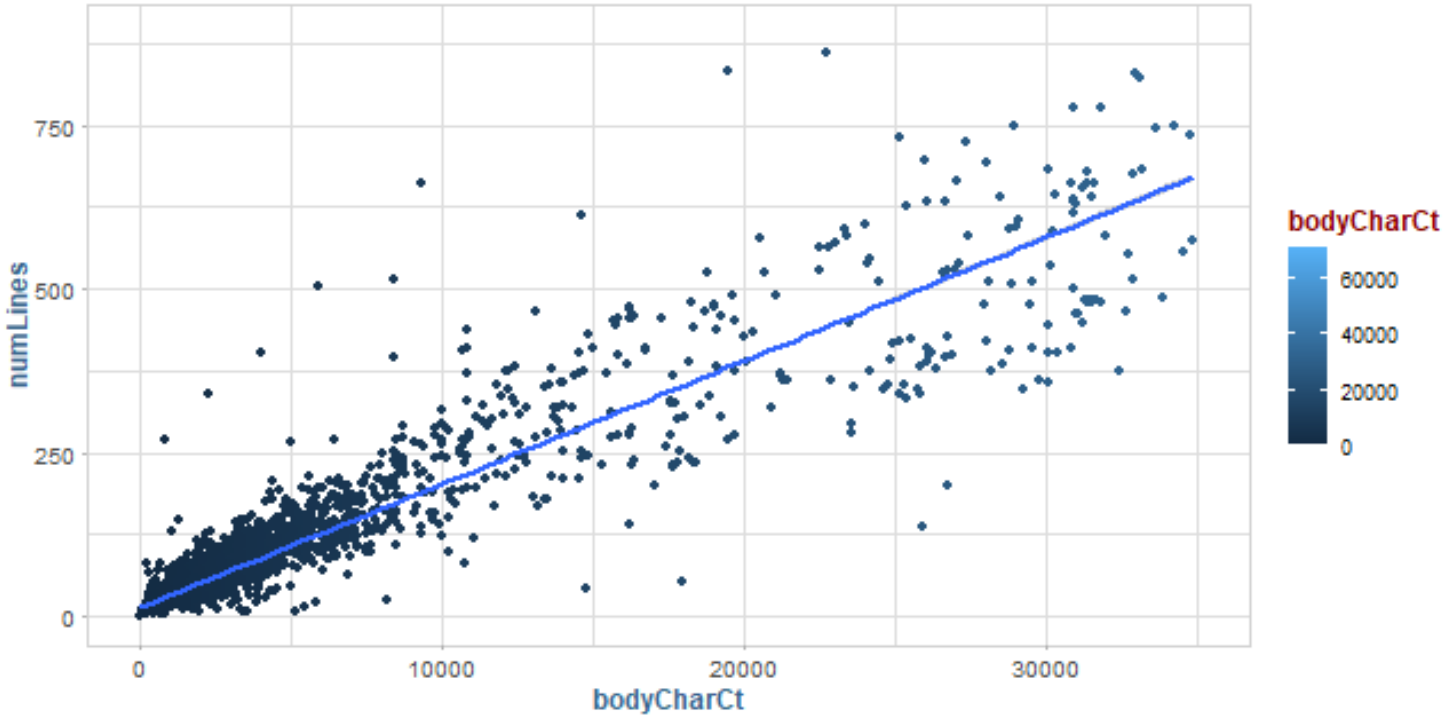
```
long_lines <- head(sort(emailDF$numLines, decreasing = T), 10)

rem <- which(emailDF$numLines %in% long_lines)
```

```
ggplot(emailDF[-rem, ], aes(bodyCharCt, numLines)) +
   geom_point(aes(col = bodyCharCt)) +
   geom_smooth(method = "lm") +
   scale_x_continuous(lim = c(0, 35000))
```

```
`geom_smooth()` using formula 'y ~ x'
```

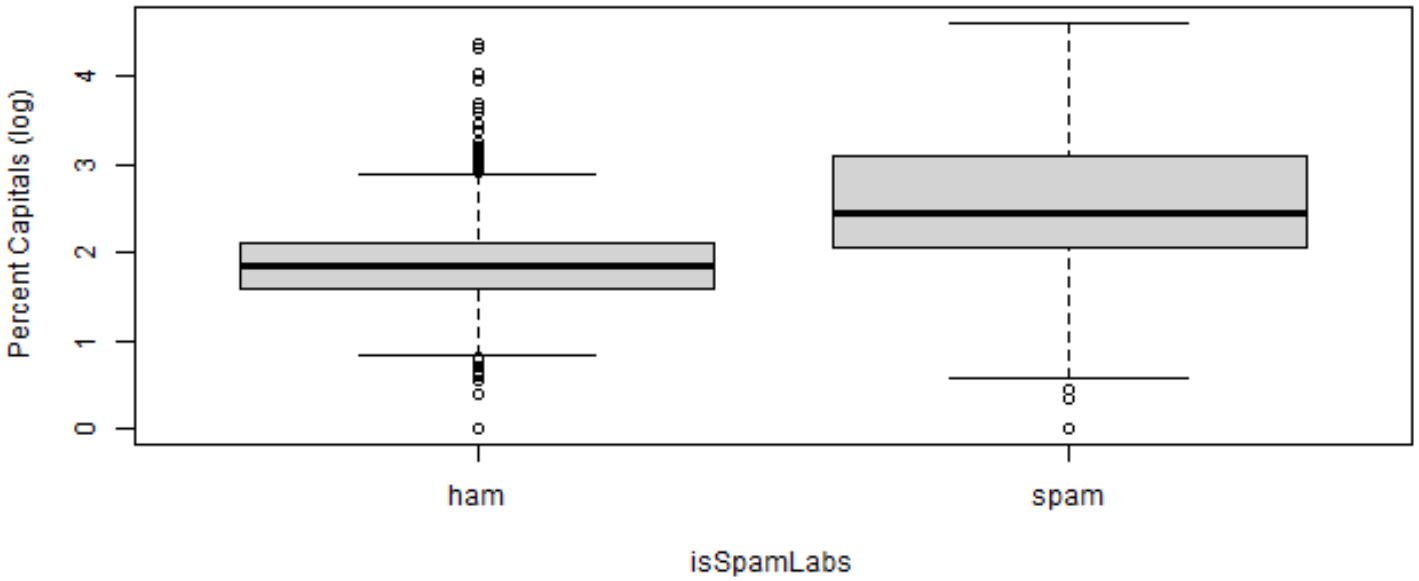
```
Warning: Removed 9 rows containing non-finite values (stat_smooth).
```

```
Warning: Removed 9 rows containing missing values (geom_point).
```



Exploring the email Feature Set

```
percent <- emailDF$perCaps
isSpamLabs <- factor(emailDF$isSpam, labels = c("ham", "spam"))
boxplot(log(1 + percent) ~ isSpamLabs,
        ylab = "Percent Capitals (log)")
```

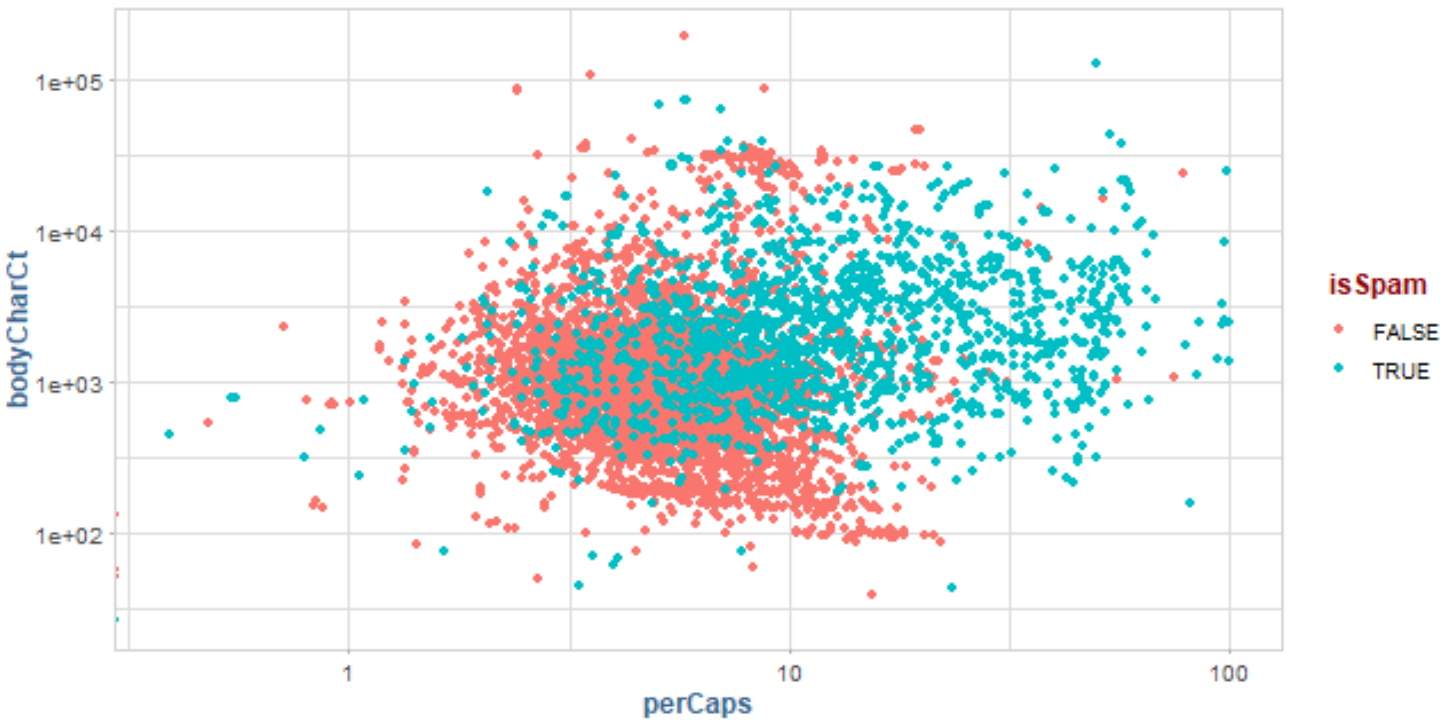



```
ggplot(emailDF, aes(perCaps, bodyCharCt, col = isSpam)) +
   geom_point() +
   scale_y_log10() +
   scale_x_log10()
```

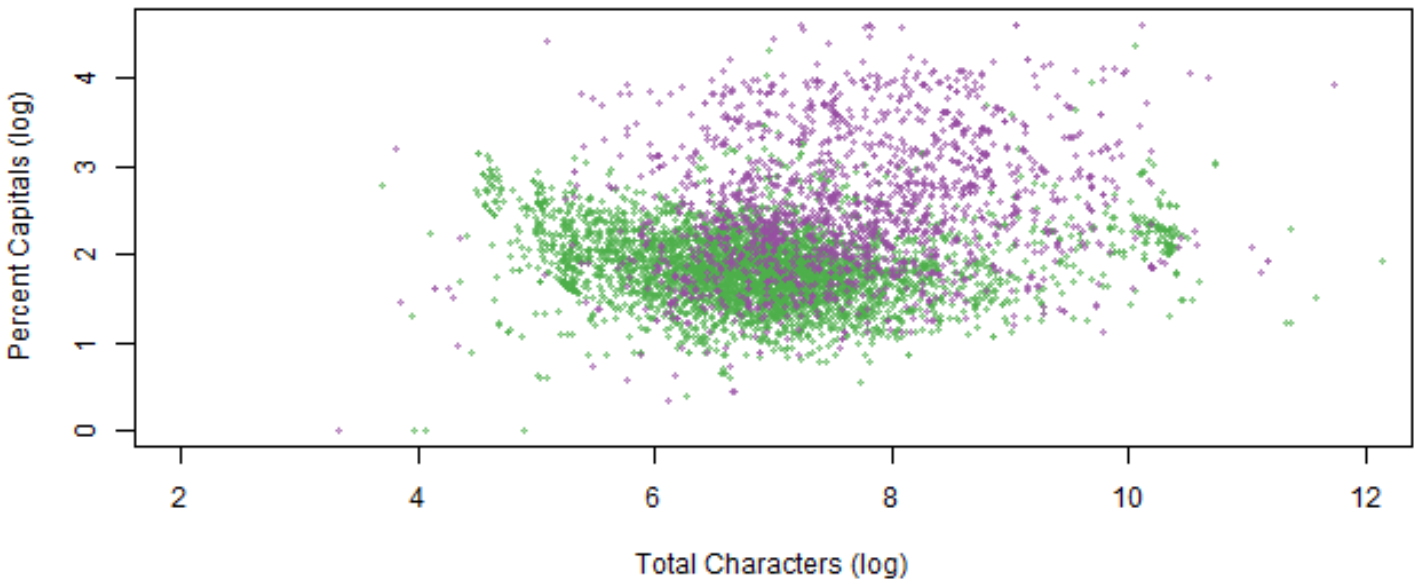
```
Warning: Transformation introduced infinite values in continuous y-axis
```

```
Warning: Transformation introduced infinite values in continuous x-axis
```

```
Warning: Removed 1 rows containing missing values (geom_point).
```



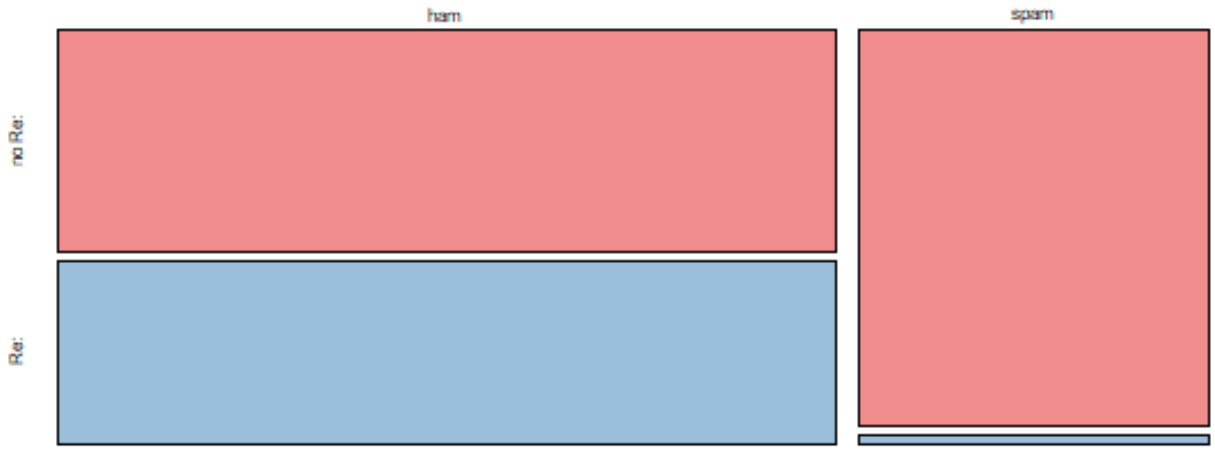

```
colI <- c("#4DAF4A80", "#984EA380")
logBodyCharCt <- log(1 + emailDF$bodyCharCt)
logPerCaps <- log(1 + emailDF$perCaps)
plot(logPerCaps ~ logBodyCharCt, xlab = "Total Characters (log)",
     ylab = "Percent Capitals (log)",
     col = colI[1 + emailDF$isSpam],
     xlim = c(2, 12), pch = 19, cex = 0.5)
```




```
table(emailDF$numAtt, isSpamLabs)
```

```
   isSpamLabs
     ham spam
  0 4010 1713
  1  183  177
  2    7    5
  4    0    1
  5    1    2
```

```
colM <- c("#E41A1C80", "#377EB880")
isRe <- factor(emailDF$isRe, labels = c("no Re:", "Re:"))
mosaicplot(table(isSpamLabs, isRe), main = "",
           xlab = "", ylab = "", color = colM)
```



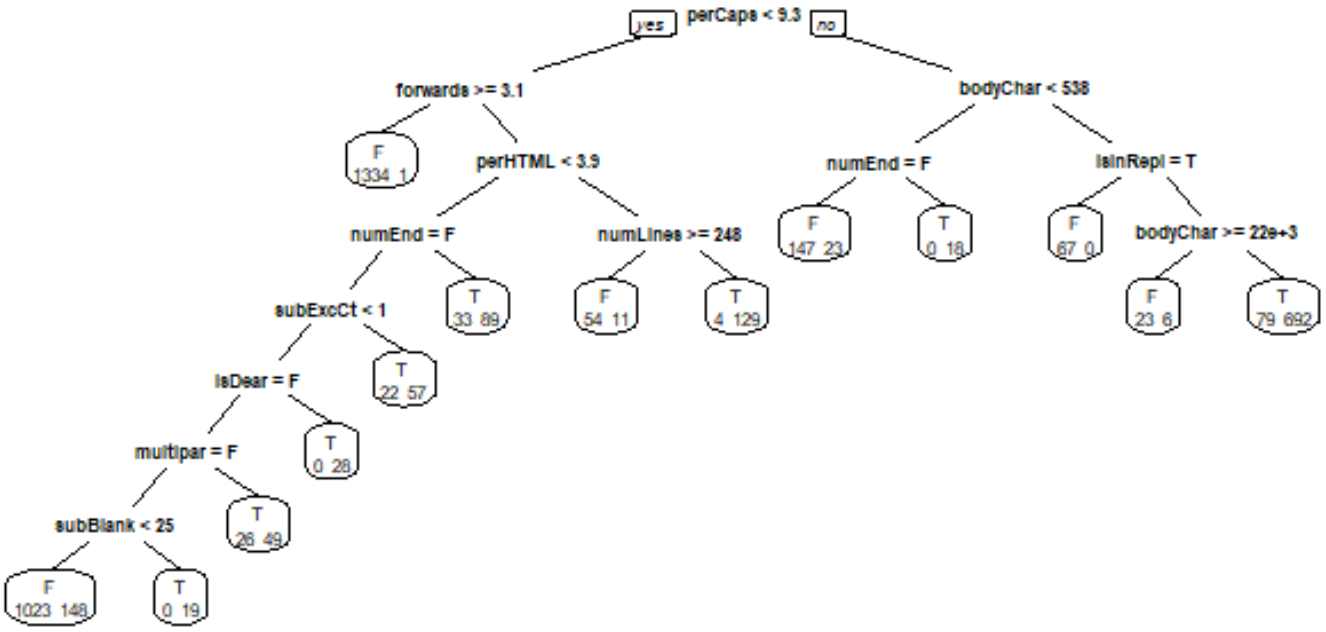

```
fromNE <- factor(emailDF$numEnd, labels = c("No #", "#"))
mosaicplot(table(isSpamLabs, fromNE), color = colM,
           main = "", xlab = "", ylab = "")
```

Fitting Recursive Partition

```
setupRpart <- function(data) {
   logicalVars <- which(sapply(data, is.logical))
   facVars <- lapply(data[, logicalVars],
                     function(x) {
                        x = as.factor(x)
                        levels(x) = c("F", "T")
                        x
                     })
   cbind(facVars, data[, - logicalVars])
}

emailDFrp <- setupRpart(emailDF)
```

```
set.seed(418910)

numSpam <- sum(isSpam)
numHam <- numEmail - numSpam

testSpamIdx <- sample(numSpam, size = floor(numSpam/3))
testHamIdx <- sample(numHam, size = floor(numHam/3))

testDF <- rbind( emailDFrp[ emailDFrp$isSpam == "T", ][testSpamIdx, ],
                 emailDFrp[ emailDFrp$isSpam == "F", ][testHamIdx, ])

trainDF <- rbind( emailDFrp[ emailDFrp$isSpam == "T", ][-testSpamIdx, ],
                  emailDFrp[ emailDFrp$isSpam == "F", ][-testHamIdx, ])
```

```
rpartFit <- rpart(isSpam ~ ., data = trainDF, method = "class")

prp(rpartFit, extra = 1)
```




```
predictions <- predict(rpartFit,
                       newdata = testDF[, names(testDF) != "isSpam"],
                       type = "class")

predsForHam <- predictions[ testDF$isSpam == "F" ]
summary(predsForHam)
```

```
   F    T NA's
1294   95 1099
```

```
sum(predsForHam == "T", na.rm = T) / length(predsForHam)
```

```
[1] 0.03818328
```

```
predsForSpam <- predictions[ testDF$isSpam == "T" ]
sum(predsForSpam == "F", na.rm = T) / length(predsForSpam)
```

```
[1] 0.05500869
```

```
args(rpart.control)
```

```
function (minsplit = 20L, minbucket = round(minsplit/3), cp = 0.01,
    maxcompete = 4L, maxsurrogate = 5L, usesurrogate = 2L, xval = 10L,
    surrogatestyle = 0L, maxdepth = 30L, ...)
NULL
```

```
complexityVals <- c(seq(0.00001, 0.0001, length = 19),
                    seq(0.0001, 0.001, length = 19),
```

```
                    seq(0.001, 0.005, length = 9),
                    seq(0.005, 0.01, length = 9))

fits <- lapply(complexityVals, function(x) {
   rpartObj <- rpart(isSpam ~ ., data = trainDF,
                     method = "class",
                     control = rpart.control(cp=x))

   predict(rpartObj,
           newdata = testDF[, names(testDF) != "isSpam"],
           type = "class")
})
```

```
spam <- testDF$isSpam == "T"
numSpam <- sum(spam, na.rm = T)
numHam <- sum(!spam, na.rm = T)

errs <- sapply(fits, function(preds) {
   typeI = sum( preds[ !spam ] == "T", na.rm = T) / numHam
   typeII = sum( preds[ spam ] == "F", na.rm = T) / numSpam
   c(typeI = typeI, typeII = typeII )
})

errs
```

```
             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
typeI  0.04967603 0.04967603 0.04967603 0.04967603 0.04967603 0.04967603
typeII 0.13694268 0.13694268 0.13694268 0.13694268 0.13694268 0.13694268
             [,7]       [,8]       [,9]      [,10]      [,11]      [,12]
typeI  0.04967603 0.04967603 0.04967603 0.04967603 0.04967603 0.04967603
typeII 0.13694268 0.13694268 0.13694268 0.13694268 0.13694268 0.13694268
            [,13]      [,14]      [,15]      [,16]      [,17]      [,18]
typeI  0.04967603 0.04967603 0.04967603 0.04967603 0.04967603 0.04967603
typeII 0.13694268 0.13694268 0.13694268 0.13694268 0.13694268 0.13694268
            [,19]      [,20]      [,21]      [,22]      [,23]      [,24]
typeI  0.04967603 0.04967603 0.04967603 0.04967603 0.04967603 0.04967603
typeII 0.13694268 0.13694268 0.13694268 0.13694268 0.13694268 0.13694268
            [,25]      [,26]      [,27]      [,28]      [,29]      [,30]
typeI  0.04967603 0.04967603 0.04967603 0.04967603 0.04607631 0.04607631
typeII 0.13694268 0.13694268 0.13694268 0.13694268 0.13694268 0.13694268
            [,31]      [,32]      [,33]      [,34]      [,35]      [,36]
typeI  0.04607631 0.04607631 0.04607631 0.04967603 0.04967603 0.04967603
typeII 0.13694268 0.13694268 0.13694268 0.11942675 0.11942675 0.11942675
            [,37]      [,38]      [,39]      [,40]      [,41]      [,42]
typeI  0.04967603 0.04967603 0.04967603 0.05111591 0.05111591 0.05327574
```

```
typeII 0.11942675 0.11942675 0.11942675 0.10668790 0.10668790 0.10668790
            [,43]      [,44]      [,45]      [,46]      [,47]      [,48]
typeI  0.05543557 0.05327574 0.05975522 0.05975522 0.05831533 0.05831533
typeII 0.10509554 0.12101911 0.11146497 0.11146497 0.11464968 0.11464968
            [,49]      [,50]     [,51]     [,52]      [,53]      [,54]
typeI  0.05831533 0.05831533 0.0662347 0.0662347 0.06839453 0.06839453
typeII 0.11464968 0.11464968 0.1178344 0.1512739 0.15127389 0.15127389
            [,55]      [,56]
typeI  0.06839453 0.06839453
typeII 0.15127389 0.15127389
```

```
err_df <- data.table(t(errs))
```

Further Analysis

1.)

We hand-selected email to belong to the sample set in the *sampleEmail*. Instead of this approach, use the sample function to choose messages at random for the sample. Be sure to take files from all 5 directories of the email.

```
dir <- directories[1]

dir_lens <- sapply(directories, function(dir) length(list.files(dir)))

new_sample <- sapply(directories[ dir_lens > 1 ], function(dir) {
  file_list <- list.files(dir)

  n <- length(file_list)

  idx <- sample(1:n, 30)

  return(file_list[idx])
})

new_sample
```

```
      D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham
 [1,] "2414.9762e257ad9e6b6633dba8432ffb90de"
 [2,] "00110.1e36beebd2dffe60b0d8f68d82bde52c"
 [3,] "0930.e6b90edda75a110d7cc7335c110cfa1c"
 [4,] "0585.b4410217f9fefbb9a9c18df5a4aa621a"
 [5,] "1445.57f9856f348cda1656331372731701eb"
 [6,] "0242.a02f8a0ce9077130c10d33db2a16ec36"
 [7,] "01863.bd70d6cfad21b043c84dab8e1e86e2be"
 [8,] "02088.17edeee5193df341ff427a8b9b20aadf"
```

```
 [9,] "02422.cdeb9f1dc58b063c7d8b14e1f5cd66d3"
[10,] "2525.9559a3f0d4d17d33379acdf5bf356435"
[11,] "2528.8b6dd74e9b8d04f5f1338bcec51b0b11"
[12,] "02055.80f7eff41824e0337e453a988ceda994"
[13,] "0093.0c71febfdf6f3acbc4d0c76b777a8530"
[14,] "2073.c141c0e2fb130f861a8eb1414d297b08"
[15,] "00043.d2673a72d215cbdd747dc98cde41fbd2"
[16,] "00242.640f27e47a5754dbf4893781ce156a75"
[17,] "0736.0c8647e849c1d900b6af3a6c7024752d"
[18,] "1612.17083808deba0447726df856cbb574aa"
[19,] "1319.58000a90a0ce3ea98762d31df028af02"
[20,] "2310.5c05ac6e4c2b473ef5dec9fe794ec853"
[21,] "02001.2c618fdfdfa2ea01d0a5b6dc936942fa"
[22,] "1020.7fdf27321484091bd72c08886ab53262"
[23,] "00727.ea5bab335cc61c1d3d85a8566232f5e8"
[24,] "0259.434a3208757e9738f7af6a004f42c5f1"
[25,] "02383.833224f74156975eb4364e6a4ddf9a0a"
[26,] "01891.ea821f774736a6a9b155fe0dd4a53ad1"
[27,] "1688.e2ef9f60c860b46bf3afcf1178378c3b"
[28,] "2546.dd6480d3f87f7d525f797d17ac1a0bc8"
[29,] "02378.ff357ba03232ac47ae4cbdf1a770cfc4"
[30,] "0057.be5e34dcebd922928045634015e3ed78"
      D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham_2
 [1,] "00720.b32e7900b189a55cf7207e9633f5c437"
 [2,] "00793.b4ae3b05f6b8dfc24f8a92ab759ba54d"
 [3,] "00263.84321935c6f5a34e6a124bbd64b9b5c7"
 [4,] "00340.22ecbee41fb4da91afa69f932cb27443"
 [5,] "00543.5ba1d9e8383ecbba9dc5733e9822ee1b"
 [6,] "00963.5480cd0c553b673d4b2a5dd8775bc858"
 [7,] "00840.49f8f5847e553c654524fcf531212b1a"
 [8,] "00548.9df9bd35a18874dcf39ec227a063b847"
 [9,] "00606.aff1067a665502934f28d2494bf9ed29"
[10,] "00390.faa916d56707c7ea2a82dca0cbabaec9"
[11,] "00121.4c398f0106848ae9f9d3462c2296de17"
[12,] "00561.3e703cebc65221a731b98dc16d963d94"
[13,] "00426.6cd93f7b4c74456e414f1f01fec6b05f"
[14,] "01142.fbca515af7491a2cb7eec15d7011fd7a"
[15,] "00636.934784d4ed76cc8ad094ac8261dd9bcf"
[16,] "00160.b7ad2878346c13c7726f872d632c42b5"
[17,] "00575.4fea6123edb74361f0cf88c7621174e0"
[18,] "00670.cf4700dea8b59597f608d0e7062e605a"
[19,] "00484.f7052aa77491ee5979a55c16eae22422"
[20,] "00678.7562e626b536eb5c1534ec1de6cb8259"
[21,] "01060.95d3e0a8c47b33d1533f18ac2c60c81a"
[22,] "00056.6647a720da7dad641f4028c9f6fbf4e5"
```

```
[23,] "00724.c5a687a60e45b7e78ec560a3eadffdee"
[24,] "01348.097c29b68042a1d73710d49021577739"
[25,] "00299.f5ee5d9a3056c28135db57935818e138"
[26,] "00269.5e79c797bc756cc555e6877c5fbefc04"
[27,] "01224.9a1ab58ee0dfbe03561e80fb9c84071e"
[28,] "00797.67afafcb6abaa29b1f92fafa4e153916"
[29,] "00873.591767bb0a3b3c5f3fe4a7b8d712bb6e"
[30,] "01195.e7834b72cb6abc842f6d61f8cb08e346"
      D:/Projects/Statistical-Computing/Case Studies/datasets/spam/hard_ham
 [1,] "0143.078ced89f2de25e9423f53798e6e3a70"
 [2,] "00065.84b8fba96e680358bbd2c961fe356982"
 [3,] "0075.b233a905fb7f4cc8f9f878aeb9f7b36d"
 [4,] "00159.4ebed46c00f57c37a36d66184a08052c"
 [5,] "00245.ad6a6e01bee8f1d84541fb127a86e31f"
 [6,] "00036.b2641f0e9ff64695cf368088640bee0c"
 [7,] "00155.b84fe135fb77395651a5b88a1f808cf9"
 [8,] "0097.8b611c66f47a5dfc476a015fc2395e12"
 [9,] "00223.14b06feeb8b03fed4e272140b8ed95f0"
[10,] "0007.7f2ea3a532284cff3321e5ba159cdb50"
[11,] "0003.0aa92b5f121c27c6e094fd89c6c89448"
[12,] "0162.5258f5bbb6d86187bbd7b58851ce3b06"
[13,] "00140.1064ebb32a5625a958a6e38ffe9e061c"
[14,] "0160.6afa0b4b6fd169beb74a46759a413fda"
[15,] "00218.c8c8534c14d8e1b43c43a7565a16f28a"
[16,] "00197.c7483488867fe74e7444441283549ae8"
[17,] "00007.d24e99a602ee7fb442714c0d448cd08e"
[18,] "00144.02ec2c8f4d1c6b004c9eba1bf124f8e3"
[19,] "0056.5a2d7682765ade8f025450dfc82be013"
[20,] "00117.cf6312eae6441d25bef2ecfc39cc4acb"
[21,] "00082.a405e76faf9463d464229306b1e0c93f"
[22,] "00139.8164b7e486cc17d8f2c921f99e05ed10"
[23,] "00131.fea96653807a20ab7a910705f7adc6c0"
[24,] "0164.82c660c5e1f778f96a4e36ab9aec323b"
[25,] "00135.fc6adca276535ab323a5afc4cec23256"
[26,] "0220.eb242d1945f31e1c05ae1df9b3bf2537"
[27,] "0224.62765917ab1fc98fe9e118aef0894765"
[28,] "0238.7c7d6921e671bbe18ebb5f893cd9bb35"
[29,] "0052.ae8ff272d7ef31c406cad02f476fefed"
[30,] "00168.f8f56df10d37e1b1a50747cf9708e8b4"
      D:/Projects/Statistical-Computing/Case Studies/datasets/spam/spam
 [1,] "0284.cfe6e278b87c3e9b6abf6cf6a16bf708"
 [2,] "00241.c28ade5771085a8fddd054a219566b7c"
 [3,] "0426.2002be3b0195b54596a5e7fd7d7561d5"
 [4,] "0010.7f5fb525755c45eb78efc18d7c9ea5aa"
 [5,] "0396.8ea0610e30c94adefd9b3489df436ad9"
```

```
 [6,] "0176.70022adaab1a9dfe64ae7588ffa5add9"
 [7,] "0254.02daa37a4255a78f2f224f3cd2f8fa99"
 [8,] "0461.27302a2e94d8948f8a81a7d4c8566cf0"
 [9,] "00273.0c7d73771d79e84e2aab8c909c5bb210"
[10,] "00278.b62c5fc23a2f87760696cb9fa51f073c"
[11,] "00200.bacd4b2168049778b480367ca670254f"
[12,] "00079.cc3fa7d977a44a09d450dde5db161c37"
[13,] "00279.1d58a13e343c1e53aca2ed2121a3f815"
[14,] "cmds"
[15,] "00351.fd1b8a6cd42e81125fb38c2660cd9317"
[16,] "0412.4e18b948471feca1fa1610ce7c1259a2"
[17,] "00207.0b71ac81a360455c1514f5872564b1e1"
[18,] "0016.f9c349935955e1ccc7626270da898445"
[19,] "00046.e0fd04360622dbe9250380447f6465cc"
[20,] "0001.bfc8d64d12b325ff385cca8d07b84288"
[21,] "00134.9f41f4111a33dc1efca04de72e1a105a"
[22,] "00194.767c323b4ae7a4909397e42cbd0c56a4"
[23,] "00350.c2658f17a328efdf045b38ab38db472f"
[24,] "0487.b57549dc531f50c1ff1e3356bc38b390"
[25,] "00284.4cdf4c9e9404c79c85ab5ac12ce39e85"
[26,] "00392.ffefdd973d6b1bf1243937030e3bd07f"
[27,] "00102.fb09d2f978a271fba5a3ffc172003ed9"
[28,] "0179.3a4c735c7c1e494f4e7a7b9465043280"
[29,] "0212.9a9f009a6d601e2e34c1b95353983352"
[30,] "0134.83a63d7a1589ba4cd6aefe20c8e6385f"
      D:/Projects/Statistical-Computing/Case Studies/datasets/spam/spam_2
 [1,] "00955.0e418cf2dca0e0ac90fcaf35f5cedbc3"
 [2,] "00275.87c74dc27e397ccd3b2b581bbefef515"
 [3,] "00438.cf76c0c71830d5e8ddec01a597f149a5"
 [4,] "00879.ef1461ca38091f6d494c58d09b0627f0"
 [5,] "00263.32b258c4cc08d235b2ca36fc16074f08"
 [6,] "00740.ce4777381c2bc6bee30bef6bd274233f"
 [7,] "01369.8ea24235c6c50337d9dcd234e61a5132"
 [8,] "00046.96a19afe71cd6f1f14c96293557a49ff"
 [9,] "00727.45ac8c0efbb22514a075b99e1c57422e"
[10,] "00669.790cde659c7d18535eb46cfa4398458d"
[11,] "00664.c4f198903588cdc4af385772bb580d90"
[12,] "00776.22f3f3942932c3d3b6e254bcab9673d1"
[13,] "00906.bd0b0986deaf717b1f1a689fd950b97c"
[14,] "00463.0bc4e08af0529dd773d9f10f922547db"
[15,] "00013.372ec9dc663418ca71f7d880a76f117a"
[16,] "00895.d7895e10504f34149655062fe20d5174"
[17,] "00643.d177c04238b4299813b7d8cca9fb2f18"
[18,] "00117.9f0ba9c35b1fe59307e32b7c2c0d4e61"
[19,] "00848.1fe2e3c6535ebd22e457a3de8e5508b9"
```

```
[20,] "00537.c61f1e424853d045bd87415405cf8cfe"
[21,] "01317.2fb1c15091162a0a83cb7020e45e8de6"
[22,] "00877.98d4bfaa6f6c0a303d80544b39d7dc66"
[23,] "00447.32e588c3a1d8888d737f360f825713b8"
[24,] "00613.047cf0bcc74950bd9e1eaf8d336c385c"
[25,] "00297.6278795e285879c8623bf7ec329b966e"
[26,] "00970.4166b8fab10bbf38aa782c311a70ee6b"
[27,] "00054.58b5d10599e5e7c98ce1498f2ba3e42c"
[28,] "00578.c65d716f2fe3db8abc5deb3cbc35029c"
[29,] "00330.97460a01c80eb8ef3958118baf379c93"
[30,] "01002.406c1c709e49cb740f0ce36ebf2d5c78"
```

2.) In the text mining approach to detecting spam we ignored all attachments in creating the set of words belonging to a message (see Section 3.5.2). Write a function to extract words from any plain text or HTML attachment and include these words in the set of a message's words. Try to reuse *findMsg()* function and modify the *dropAttach()* function to accept an additional parameter that indicates whether or not the words in the attachments are to be extracted.

```
index <- hasAttach[2]

boundary <- getBoundary(headerList[[index]])
body <- sampleSplit[[index]]$body
body
```

```
 [1] "(This list is sponsored by Ironclad Networks http://www.ironclad.net.au/)"
 [2] ""
 [3] "This is a multi-part message in MIME format."
 [4] ""
 [5] "------=_NextPart_000_00C1_01C25017.F2F04E20"
 [6] "Content-Type: text/plain;"
 [7] "\tcharset=\"Windows-1252\""
 [8] "Content-Transfer-Encoding: quoted-printable"
 [9] ""
[10] "I'm using Simple DNS from JHSoft.  We support only a few web sites and ="
[11] "I'd like to swap secondary services with someone in a similar position."
[12] ""
[13] "We have a static IP, DSL line and a 24/7 set of web, SQL, mail and now a ="
[14] "DNS server.  As I said, we are hosting about 10 web sites, web and DNS ="
[15] "traffic is almost nothing.  Everything is on lightly loaded APC battery ="
[16] "backups so we are very seldom down."
[17] ""
[18] "I'd like to swap with someone also using Simple DNS to take advantage of ="
```

```
[19] "the trusted zone file transfer option."
[20] ""
[21] ""
[22] ""
[23] "Bob Musser"
[24] "Database Services, Inc."
[25] "Makers of:"
[26] "   Process Server's Toolbox"
[27] "   Courier Service Toolbox"
[28] "BobM@dbsinfo.com"
[29] "www.dbsinfo.com"
[30] "106 Longhorn Road"
[31] "Winter Park FL 32792"
[32] "(407) 679-1539"
[33] ""
[34] ""
[35] ""
[36] "------=_NextPart_000_00C1_01C25017.F2F04E20"
[37] "Content-Type: text/html;"
[38] "\tcharset=\"Windows-1252\""
[39] "Content-Transfer-Encoding: quoted-printable"
[40] ""
[41] "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">"
[42] "<HTML><HEAD>"
[43] "<META http-equiv=3DContent-Type content=3D\"text/html; ="
[44] "charset=3Dwindows-1252\">"
[45] "<META content=3D\"MSHTML 6.00.2716.2200\" name=3DGENERATOR>"
[46] "<STYLE></STYLE>"
[47] "</HEAD>"
[48] "<BODY bgColor=3D#ffffff>"
[49] "<DIV><FONT size=3D2>I'm using Simple DNS from JHSoft.  We support ="
[50] "only a few=20"
[51] "web sites and I'd like to swap secondary services with someone in a ="
[52] "similar=20"
[53] "position.</FONT></DIV>"
[54] "<DIV><FONT size=3D2></FONT> </DIV>"
[55] "<DIV><FONT size=3D2>We have a static IP, DSL line and a 24/7 set of web, ="
[56] "SQL, mail=20"
[57] "and now a DNS server.  As I said, we are hosting about 10 web ="
[58] "sites, web=20"
[59] "and DNS traffic is almost nothing.  Everything is on lightly loaded ="
[60] "APC=20"
[61] "battery backups so we are very seldom down.</FONT></DIV>"
[62] "<DIV><FONT size=3D2></FONT> </DIV>"
[63] "<DIV><FONT size=3D2>I'd like to swap with someone also using Simple DNS ="
```

```
[64] "to take=20"
[65] "advantage of the trusted zone file transfer option.</FONT></DIV>"
[66] "<DIV><FONT size=3D2></FONT> </DIV>"
[67] "<DIV><FONT size=3D2></FONT> </DIV>"
[68] "<DIV><FONT size=3D2></FONT> </DIV>"
[69] "<DIV><FONT size=3D2>Bob Musser<BR>Database Services, Inc.<BR>Makers=20"
[70] "of:<BR>   Process Server's Toolbox<BR>   Courier ="
[71] "Service=20"
[72] "Toolbox<BR><A ="
[73] "href=3D\"mailto:BobM@dbsinfo.com\">BobM@dbsinfo.com</A><BR><A=20"
[74] "href=3D\"http://www.dbsinfo.com\">www.dbsinfo.com</A><BR>106 Longhorn ="
[75] "Road<BR>Winter=20"
[76] "Park FL 32792<BR>(407) 679-1539</FONT></DIV>"
[77] "<DIV> </DIV>"
[78] "<DIV><FONT size=3D2></FONT> </DIV></BODY></HTML>"
[79] ""
[80] "------=_NextPart_000_00C1_01C25017.F2F04E20--"
[81] ""
[82] ""
[83] "--"
[84] "To Unsubscribe: <dns-swap-off@lists.ironclad.net.au>"
[85] "Sponsor & Host: Ironclad Networks <http://www.ironclad.net.au/>"
[86] ""
```

```
includeAttach <- function(body, boundary) {

   if(is.null(body)) {
      return("")
   }

   bString <- paste("--", boundary, sep = "")
   bStringLocs <- which(bString == body)

   eString <- paste("--", boundary, "--", sep = "")
   eStringLoc <- which(eString == body)

   return(msg)
}
```

3.)

The string manipulation functions in R can be used instead of regular expression functions for finding, changing and extracting substrings from strings. These functions include: *strsplit()* to divide a string up into pieces, *substr()* to extract a portion of a string, *paste()* to glue together multiple strings, and *nchar()*, which returns the number of characters in a string. Write your own version of the get boundary strings from

the Content-Type.

```
header <- sampleSplit[[6]]$header
boundaryIdx <- grep("boundary=", header)
header[boundaryIdx]
```

```
[1] "    boundary=\"==_Exmh_-1317289252P\";"
```

```
boundary <- header[ str_which(header, "boundary=") ]
pieces <- unlist(strsplit(boundary, '='))
pieces <- pieces[ str_which(pieces, ";") ]


sub(".*boundary=\"(.*)\";.*", "\\1", header[boundaryIdx])
```

```
[1] "==_Exmh_-1317289252P"
```

```
header2 <- headerList[[9]]
boundaryIdx2 <- grep("boundary=", header2)
header2[boundaryIdx2]
```

```
                                          Content-Type
"multipart/alternative; boundary=Apple-Mail-2-874629474"
```

```
sub('.*boundary="(.*)";.*', "\\1", header2[boundaryIdx2])
```

```
                                          Content-Type
"multipart/alternative; boundary=Apple-Mail-2-874629474"
```

```
boundary2 <- gsub('"', "", header2[boundaryIdx2])

sub(".*boundary= *(.*);?.*", "\\1", boundary2)
```

```
            Content-Type
"Apple-Mail-2-874629474"
```

```
boundary <- gsub('"', "", header[boundaryIdx])
sub(".*boundary= *(.*);?.*", "\\1", boundary)
```

```
[1] "==_Exmh_-1317289252P;"
```

```
getBoundary2 <- function(header) {
   boundary <- header[ str_which(header, "boundary=") ]
   pieces <- unlist(strsplit(boundary, '='))
   pieces <- pieces[ str_which(pieces, ";") ]
   paste("==", pieces, sep = "")
}

getBoundary(header)
```

```
[1] "==_Exmh_-1317289252P"
```

```
getBoundary2(header)
```

```
[1] "==_Exmh_-1317289252P\";"
```

4.)

Write the __dropAttach()_ function for Section 3.5.2. This funciton has two inputs, the body of a mkessage and the boundary string that marks the location of the attachments. It returns the body without its attachments. Include in the return value the lines of the body that follow the first boundary string up to the string marking the first attachment and the lines following the ending boundary string. Be sure to consider the idiosyncratic cases of no attachments and a missing ending boundary string.

```
processAttach <- function(body, contentType ) {

   boundary <- getBoundary(contentType)

   bString <- paste("--", boundary, sep = "")
   bStringLocs <- which(bString == body)

   eString <- paste("--", boundary, "--", sep = "")
   eStringLoc <- which(eString == body)

   n <- length(body)

   if(length(bStringLocs) == 2) {

      bodyContent <- body[(bStringLocs[1] + 2):(bStringLocs[2] - 1)]

      emptyLines <- which(bodyContent == "")
      bodyContent <- bodyContent[ - emptyLines]

      attachContent <- body[(bStringLocs[2] + 1):n]

      aLen <- diff(c(bStringLocs[-1], eStringLoc))
      aType <- body[bStringLocs[-1] + 1]

      if(length(aLen) == length(aType)) {
         attachments <- data.frame(aLen = aLen, aType = aType)
      } else {
         attachments <- data.frame(aLen = c(), aType = c())
      }

   } else {
      if( length(bStringLocs) == 0 ) {
```

```
         bodyContent <- body
      } else {
         bodyContent = body
      }
      attachments <- data.frame(aLen = c(), aType = c())
   }

   return(list(body = bodyContent, attachDF = attachments ))
}
```

5.)

Write the function **findMsgWords()** of Section 3.5.3. This function takes as input the message body (with no attachments) and the return value is a vector of the unique words in the message. That is, we only track which words are in the message, not the number of times these words appear in the message. Consider wheather it is simpler to split the string by blanks first and then process the puncuation, digits, etc. The function should convert capital letters to lower case and drop all stop words and words that are only one letter long. A vector of stop words is avaliable in the tm package.

```
msg <- sampleSplit[[3]]$body
msg
```

```
 [1] "Man Threatens Explosion In Moscow "
 [2] ""
 [3] "Thursday August 22, 2002 1:40 PM"
 [4] "MOSCOW (AP) - Security officers on Thursday seized an unidentified man who"
 [5] "said he was armed with explosives and threatened to blow up his truck in"
 [6] "front of Russia's Federal Security Services headquarters in Moscow, NTV"
 [7] "television reported."
 [8] "The officers seized an automatic rifle the man was carrying, then the man"
 [9] "got out of the truck and was taken into custody, NTV said. No other details"
[10] "were immediately available."
[11] "The man had demanded talks with high government officials, the Interfax and"
[12] "ITAR-Tass news agencies said. Ekho Moskvy radio reported that he wanted to"
[13] "talk with Russian President Vladimir Putin."
[14] "Police and security forces rushed to the Security Service building, within"
[15] "blocks of the Kremlin, Red Square and the Bolshoi Ballet, and surrounded the"
[16] "man, who claimed to have one and a half tons of explosives, the news"
[17] "agencies said. Negotiations continued for about one and a half hours outside"
[18] "the building, ITAR-Tass and Interfax reported, citing witnesses."
[19] "The man later drove away from the building, under police escort, and drove"
[20] "to a street near Moscow's Olympic Penta Hotel, where authorities held"
[21] "further negotiations with him, the Moscow police press service said. The"
[22] "move appeared to be an attempt by security services to get him to a more"
[23] "secure location. "
```

```
[24] ""
[25] "------------------------ Yahoo! Groups Sponsor ---------------------~-->"
[26] "4 DVDs Free +s&p Join Now"
[27] "http://us.click.yahoo.com/pt6YBB/NXiEAA/mG3HAA/7gSolB/TM"
[28] "--------------------------------------------------------------------~->"
[29] ""
[30] "To unsubscribe from this group, send an email to:"
[31] "forteana-unsubscribe@egroups.com"
[32] ""
[33] " "
[34] ""
[35] "Your use of Yahoo! Groups is subject to http://docs.yahoo.com/info/terms/ "
[36] ""
[37] ""
[38] ""
```

```
findMsgWords <- function(msg) {

}

words <- unlist(sapply(msg, function(line) strsplit(line, " ")))
names(words) <- ""

n <- length(words)

msg_words <- vector(mode = "character")

for(i in 1:n)
{
   word <- tolower(words[i])

   if( word %in% exclude_word_list) {
      next
   }


}

tw <- "http://us.click.yahoo.com/pt6YBB/NXiEAA/mG3HAA/7gSolB/TM"

gsub(tw, pattern = "(//)", "")
```

```
[1] ""
```

```
exclude_word_list
```

```
  [1] "i"          "me"         "my"         "myself"     "we"
  [6] "our"        "ours"       "ourselves"  "you"        "your"
 [11] "yours"      "yourself"   "yourselves" "he"         "him"
 [16] "his"        "himself"    "she"        "her"        "hers"
 [21] "herself"    "it"         "its"        "itself"     "they"
 [26] "them"       "their"      "theirs"     "themselves" "what"
 [31] "which"      "who"        "whom"       "this"       "that"
 [36] "these"      "those"      "am"         "is"         "are"
 [41] "was"        "were"       "be"         "been"       "being"
 [46] "have"       "has"        "had"        "having"     "do"
 [51] "does"       "did"        "doing"      "would"      "should"
 [56] "could"      "ought"      "i'm"        "you're"     "he's"
 [61] "she's"      "it's"       "we're"      "they're"    "i've"
 [66] "you've"     "we've"      "they've"    "i'd"        "you'd"
 [71] "he'd"       "she'd"      "we'd"       "they'd"     "i'll"
 [76] "you'll"     "he'll"      "she'll"     "we'll"      "they'll"
 [81] "isn't"      "aren't"     "wasn't"     "weren't"    "hasn't"
 [86] "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
 [91] "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"
 [96] "cannot"     "couldn't"   "mustn't"    "let's"      "that's"
[101] "who's"      "what's"     "here's"     "there's"    "when's"
[106] "where's"    "why's"      "how's"      "a"          "an"
[111] "the"        "and"        "but"        "if"         "or"
[116] "because"    "as"         "until"      "while"      "of"
[121] "at"         "by"         "for"        "with"       "about"
[126] "against"    "between"    "into"       "through"    "during"
[131] "before"     "after"      "above"      "below"      "to"
[136] "from"       "up"         "down"       "in"         "out"
[141] "on"         "off"        "over"       "under"      "again"
[146] "further"    "then"       "once"       "here"       "there"
[151] "when"       "where"      "why"        "how"        "all"
[156] "any"        "both"       "each"       "few"        "more"
[161] "most"       "other"      "some"       "such"       "no"
[166] "nor"        "not"        "only"       "own"        "same"
[171] "so"         "than"       "too"        "very"
```

6.)