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
```

```
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")
CTloc
```

```
$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00001.7c53336b37003a928
[1] 46

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00002.9c4069e25e1ef370c
[1] 45

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00003.860e3c3cee1b42ead
[1] 42

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00004.864220c5b6930b209
[1] 30

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00005.bf27cdeaf0b8c4647
[1] 44

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00014.cb20e10b2bfcb8210
[1] 54

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00025.d685245bdc4444f44
integer(0)

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00062.009f5a1a8fa88f0b3
[1] 21
```

```
$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00063.0acbc484a73f0e0b7
[1] 17

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/0030.77828e31de08ebb58b
[1] 52

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00368.f86324a03e7ae7070
[1] 31

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00389.8606961eaeef7b921
[1] 52

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/0047.5c3e049737a2813d4a
[1] 52

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00775.0e012f37346784651
[1] 27

$`D:/Projects/Statistical-Computing/Case Studies/datasets/spam/easy_ham/00883.c44a035e7589e8307
[1] 31
```

```r
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
```

```r
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"
```

```r
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"
```

```r
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"
```

```r
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

```r
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
```

```
isSpam <- rep(c(FALSE, FALSE, FALSE, TRUE, TRUE), numMsgs)

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)
```

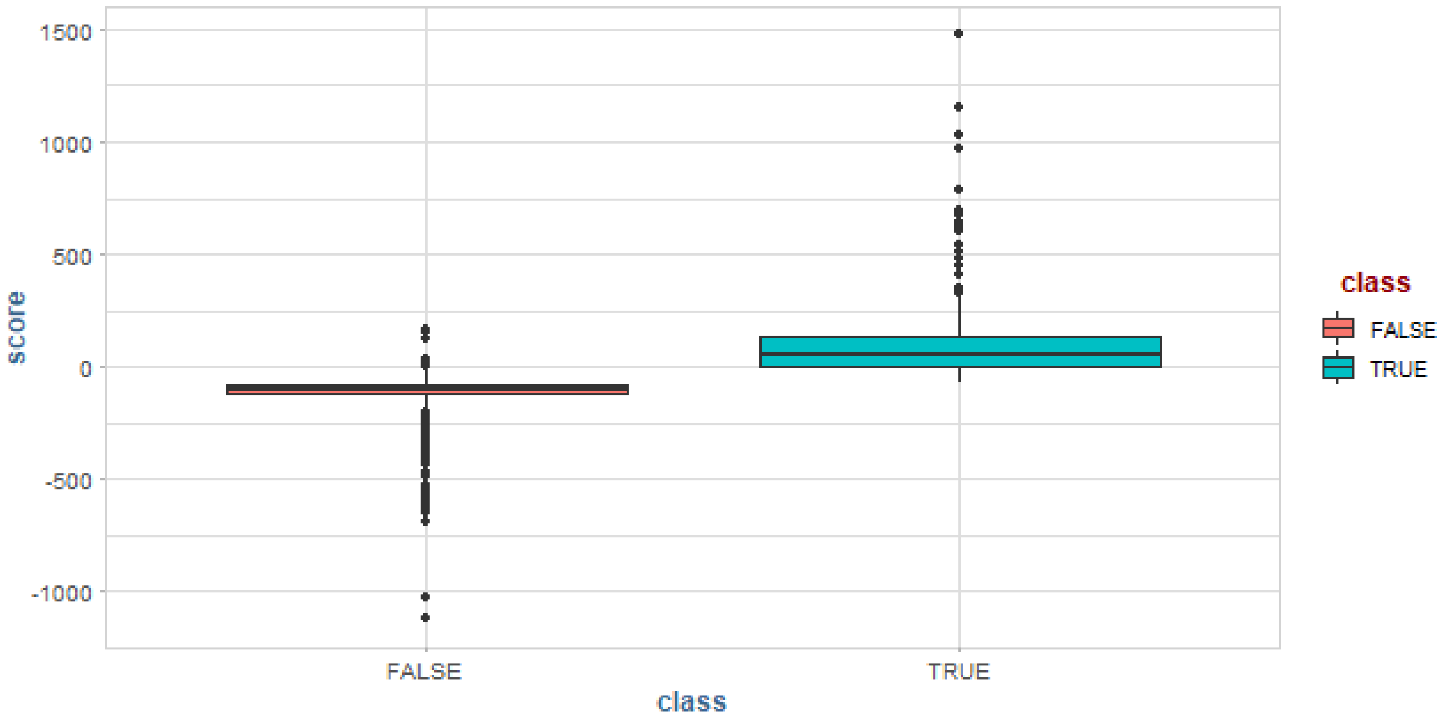
```
tapply(testLLR, testIsSpam, summary)
```

```
$`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()
```



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

typeIErrorRate(0, testLLR, testIsSpam)
```

```
[1] 0.007768666
```

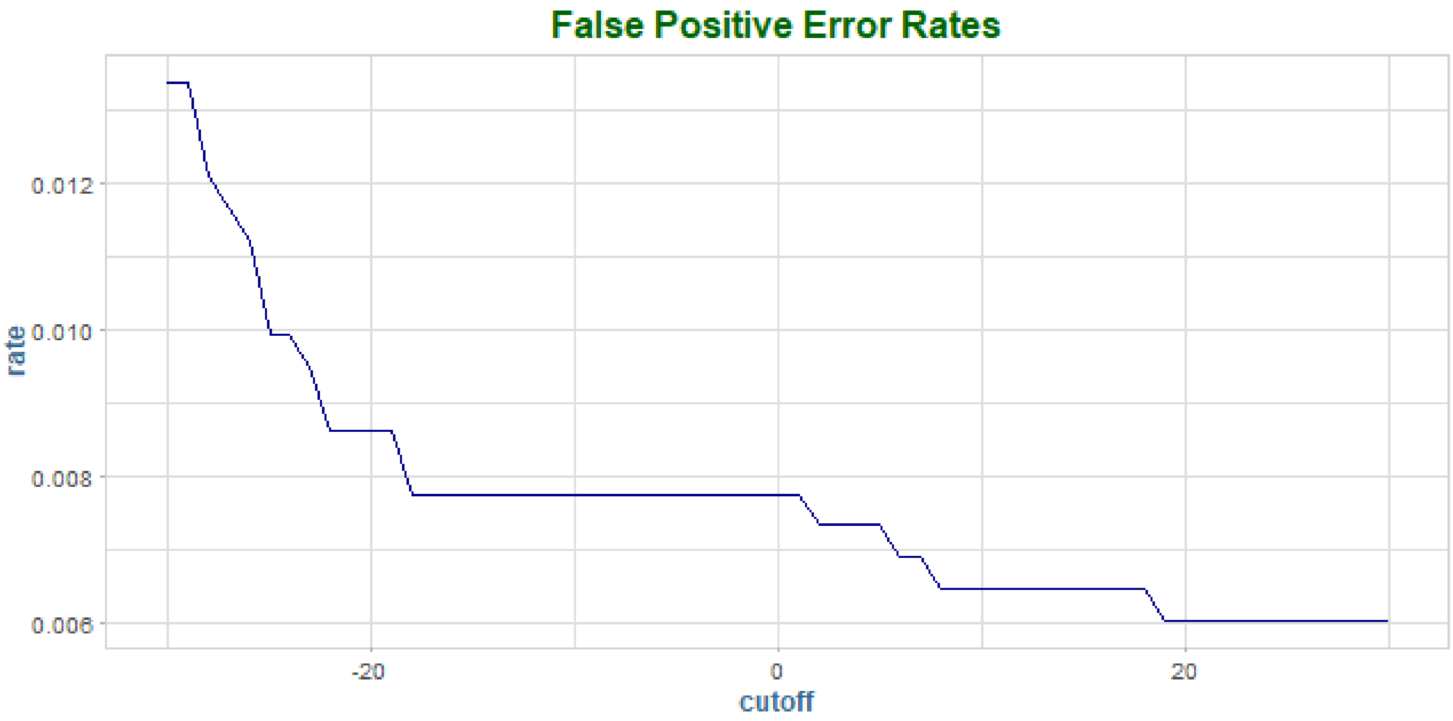
```r
typeIErrorRate(-20, testLLR, testIsSpam)
```

```
[1] 0.008631852
```

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

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" 
```

```r
length(headerVec)
```

```
[1] 36
```

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

```
[1] 26
```

```r
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)
```

```r
   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

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

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

```r
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'
```

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

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

Deriving Variables from the email Messages