Unit 5 - Separating Spam from Ham

Problem 5.1

A spam email will be displayed in the main inbox, a nuisance for the email user.
A ham email will be sent to the Junk Email folder, potentially resulting in the email user never seeing that message.

Problem 5.2

False positive
A false negative is largely a nuisance (the user will need to delete the unsolicited email). However a false positive can be very costly, since the user might completely miss an important email due to it being delivered to the spam folder. Therefore, the false positive is more costly.

Problem 5.3

A user who is particularly annoyed by spam email reaching their main inbox

Problem 5.4

A user who never checks his/her Junk Email folder

Problem 5.5

The cost of false positive results is decreased
While before many users would completely miss a ham email labeled as spam (false positive), now users will not miss an email after this sort of mistake. As a result, the cost of a false positive has been decreased.

Problem 5.6

Automatically collect information about how often each user accesses his/her Junk Email folder to infer preferences

Problem 6.1

```
wordCount = rowSums(as.matrix(dtm))
```

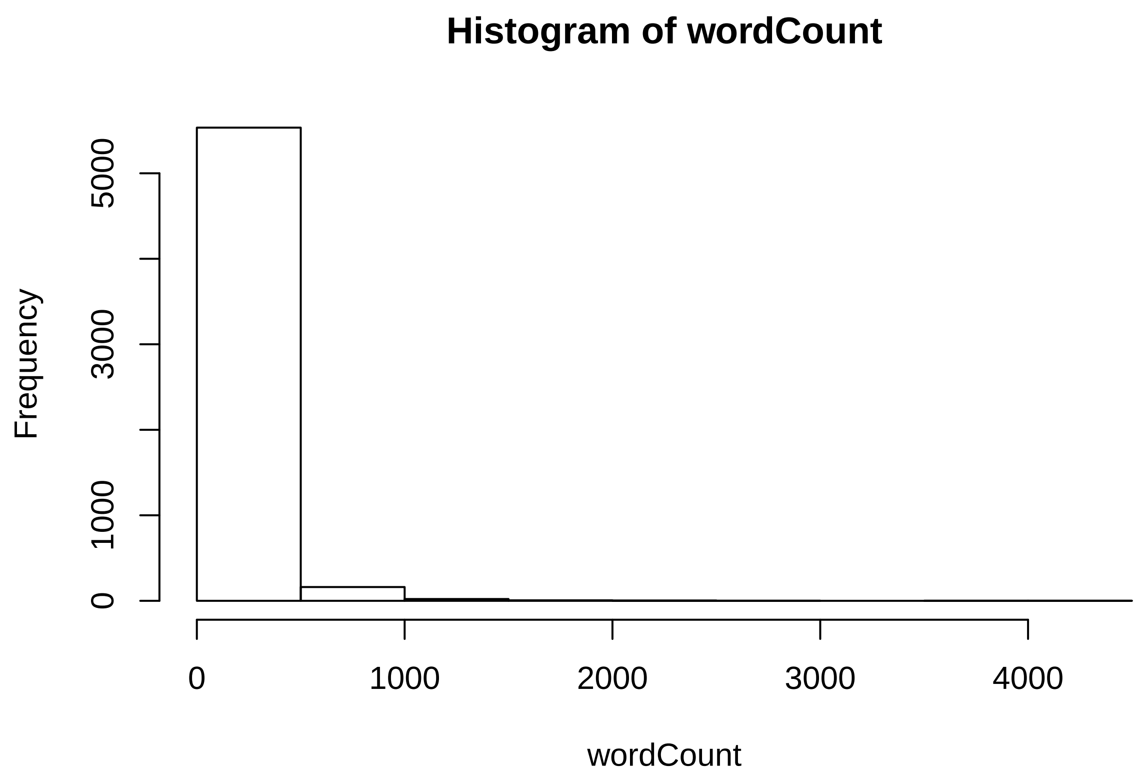
If you received an error message when running the command above, it might be because your computer ran out of memory when trying to convert dtm to a matrix. If this happened to you, try running the following lines of code instead to create wordCount (if you didn't get an error, you don't need to run these lines). This code is a little more cryptic, but is more memory efficient.

```r
library(slam)
wordCount = rollup(dtm, 2, FUN=sum)$v
```

wordCount would have only counted some of the words, but would have returned a result for all the emails correct
spdtm has had sparse terms removed, which means we have removed some of the columns but none of the rows from dtm. This means rowSums will still return a sum for each row (one for each email), but it will not have counted the frequencies of any uncommon words in the dataset. As a result, wordCount will only count some of the words.

Problem 6.2

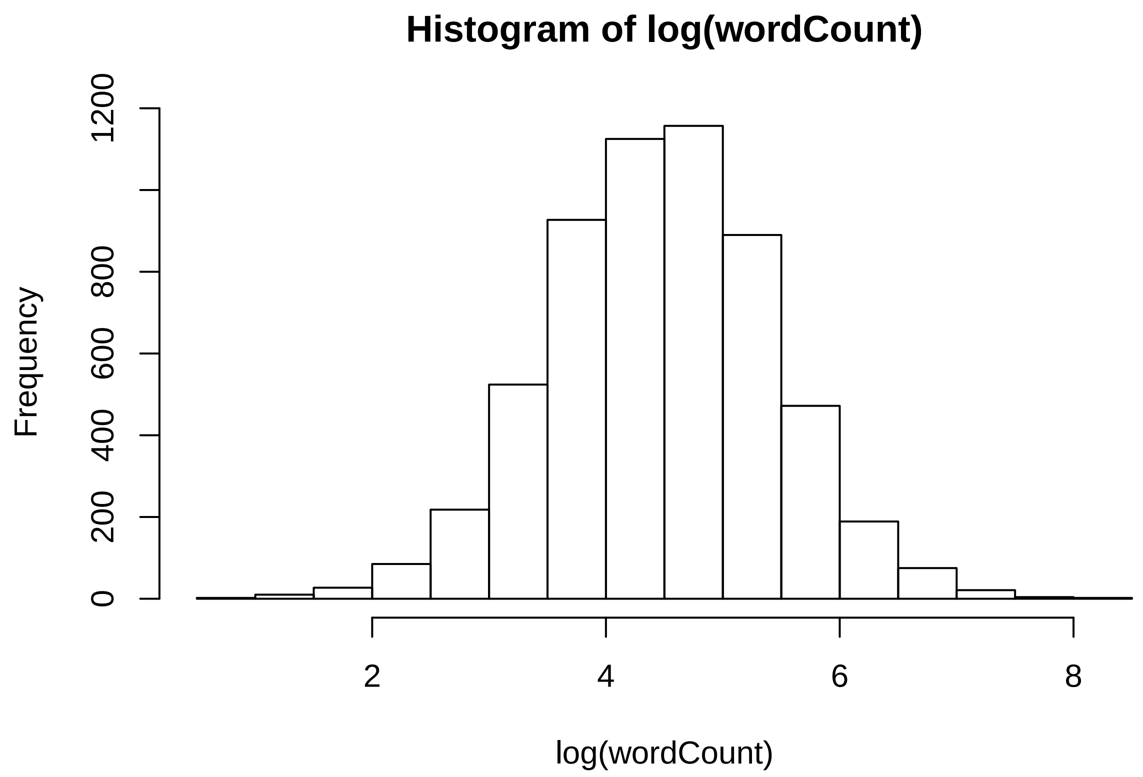
```r
hist(wordCount)
```



The data is skew right – there are a large number of small wordCount values and a small number of large values.

Problem 6.3

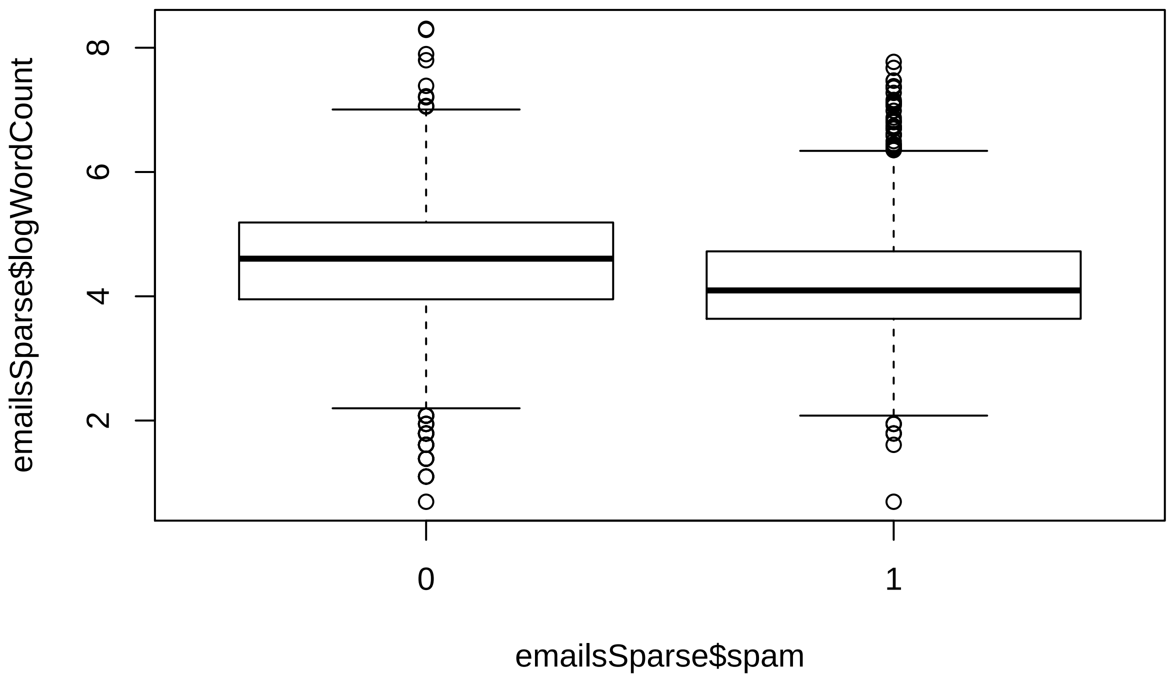
```r
hist(log(wordCount))
```



The data is not skewed – there are roughly the same number of unusually large and unusually small log(wordCount) values.

Problem 6.4

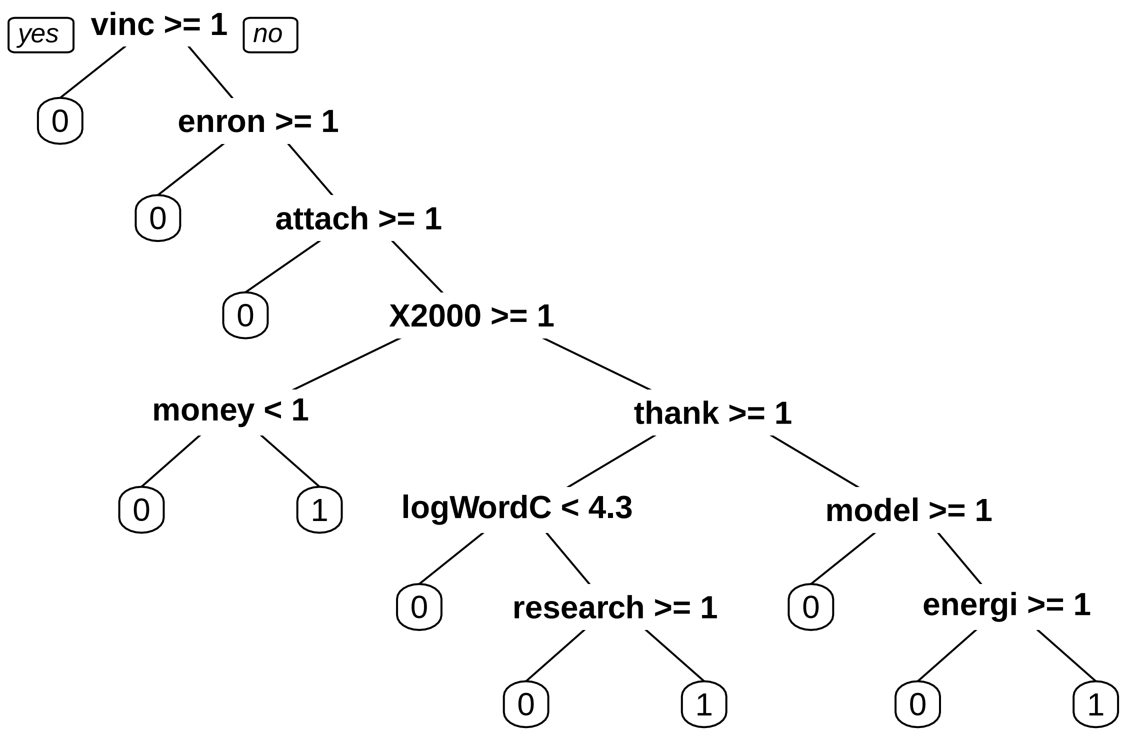
```r
emailsSparse$logWordCount <- log(wordCount)
boxplot(emailsSparse$logWordCount ~ emailsSparse$spam)
```



logWordCount is slightly smaller in spam messages than in ham messages

Problem 6.5

```
train2 <- subset(emailsSparse, spl)
test2 <- subset(emailsSparse, !spl)
spam2CART <- rpart(spam ~ ., train2, method = "class")
prp(spam2CART)
```



```r
set.seed(123)
spam2RF <- randomForest(spam ~ ., train2)
```

Yes

Problem 6.6

```r
pred2CART <- predict(spam2CART, test2)[, 2]
table(test2$spam, pred2CART > 0.5)
```

```
##
##     FALSE TRUE
##   0  1214   94
##   1    26  384
```

```r
(1214+384)/(1214+94+26+384)
```

```
## [1] 0.9301513
```

```r
predTest2RF <- predict(spam2RF, test2, type = "prob")[, 2]
```

Problem 6.7

```r
predTest2ROCRcart <- prediction(pred2CART, test2$spam)
as.numeric(performance(predTest2ROCRcart, "auc")@y.values)
```

```
## [1] 0.9582438
```

Problem 6.8

```r
table(test2$spam, predTest2RF > 0.5)
```

```
##
##     FALSE TRUE
##   0  1299    9
##   1    27  383
```

```r
(1299 + 383) / nrow(test2)
```

```
## [1] 0.9790454
```

Problem 6.9

```r
predTest2ROCRrf <- prediction(predTest2RF, test2$spam)
as.numeric(performance(predTest2ROCRrf, "auc")@y.values)
```

```
## [1] 0.9980831
```