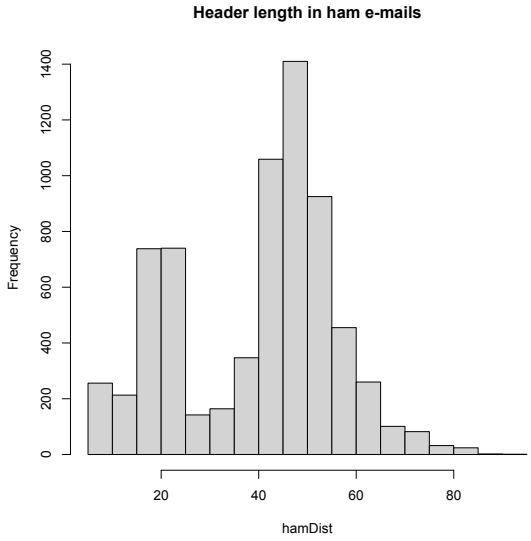
Louis Kabelka
5/5/2022
Statistical Computing

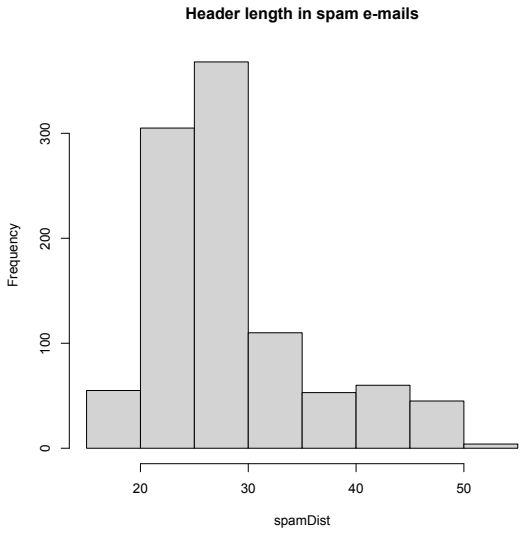
Spam Classification

Spam e-mail is fraudulent e-mail that is sent out in mass usually for malicious reasons. For user safety, it is important to be able to stop the e-mails from getting into user access. We will program algorithms that attempt to detect spam e-mail. We will use the contents of the e-mail and the e-mail metadata. Spam Assassin has provided message data of both spam and non-spam messages. Spam Assassin refers to **non-spam e-mail as ham e-mail**, which we will also do throughout this report. We will need to make some decisions on how to categorize the e-mail. For example, is worse to categorize a non-spam e-mail as spam than to categorize a spam e-mail as spam? This is the difference between type 1 & type 2 errors.
Finally, we will extend the project by looking at the e-mail header. The header contains metadata that includes the sender's e-mail address. We will try to see how based on the domain extension and the sender's username we can better observe whether or not the e-mail is spam.

1. Number of lines in Header

The e-mail header contains the encoding and routing information. The e-mail body contains the actual text of the e-mail itself. Looking at the e-mail header, we may be able to get hints on whether or not it's spam. Can the length of the e-mail header tell us if it's spam?





The ham (non-spam) e-mail headers have a mean length of 40.284 and a standard deviation of 16.184.
The spam e-mail headers have a mean length of 28.953 and a standard deviation of 7.436.
From the histograms we can see that there tend to be more lines in the headers of ham e-mails. But there is also a smaller peak in the ham e-mails that similar to the mode of the spam e-mails. So, we can't necessarily draw conclusions about e-mails with few lines. But if an e-mail has >50 lines in in the header, then it would be very unlikely that it's a spam e-mail.

2. Attachments & boundary separator

a. Attachments

Using information from the e-mail header, we can see whether or not the e-mail has attachments. If it has an attachment the header will satisfy the following conditions:
1. It will have line with "Content-Type".
2. That line will contain either "multipart" or "MULTIPART".

There are 95 files with attachments in the hard_ham directory.

b. Boundary separator

Every e-mail with attachment has a boundary separator that separate the attachment information from the e-mail.

We extracted the full boundary string from a set of e-mail with attachments. Below are the extracted boundary strings that have the boundary string only.
This means it removes any excess part from the string lines. This will definitely be removed:
1. "boundary=" before the boundary string
The following may or may not need to be removed:
2. the quotes (**""**) around the boundary string
3. the semicolon (**;**) at the end of the string
4. the blank space ( ) after the "boundary="
5. anything after the closing quote or semicolon

The following is some extraction examples of different cases:

1. Boundary tags

```
##     boundary="==_Exmh_-1317289252P";
##     boundary="----=_NextPart_000_00C1_01C25017.F2F04E20"
## Content-Type: multipart/alternative; boundary=Apple-Mail-2-874629474
##     boundary="_NextPart_1_bvfoDiTVghtoCXFdvJNKcuWblFV"
##     boundary="------------080808010909060409040405"
##     boundary="==_Exmh_1547759024P";
##     boundary="----------=_1034083278-26594-4";
##     boundary="==_Exmh_-1317289252P";
##     boundary="==_Exmh_-518574644P";
##     boundary="------------090602010909000705010009"
##     boundary="==_Exmh_-403670396P";
##     boundary="==_Exmh_-398538836P";
##     boundary="==_Exmh_-763629846P";
##     boundary="==_Exmh_-451422450P";
```

2. Result from extraction algorithm

```
> boundaries
 [1] "==_Exmh_-1317289252P"
 [2] "----=_NextPart_000_00C1_01C25017.F2F04E20"
 [3] "Apple-Mail-2-874629474"
 [4] "_NextPart_1_bvfoDiTVghtoCXFdvJNKcuWblFV"
 [5] "------------080808010909060409040405"
 [6] "==_Exmh_1547759024P"
 [7] "----------=_1034083278-26594-4"
 [8] "==_Exmh_-1317289252P"
 [9] "==_Exmh_-518574644P"
[10] "------------090602010909000705010009"
[11] "==_Exmh_-403670396P"
[12] "==_Exmh_-398538836P"
[13] "==_Exmh_-763629846P"
[14] "==_Exmh_-451422450P"
```

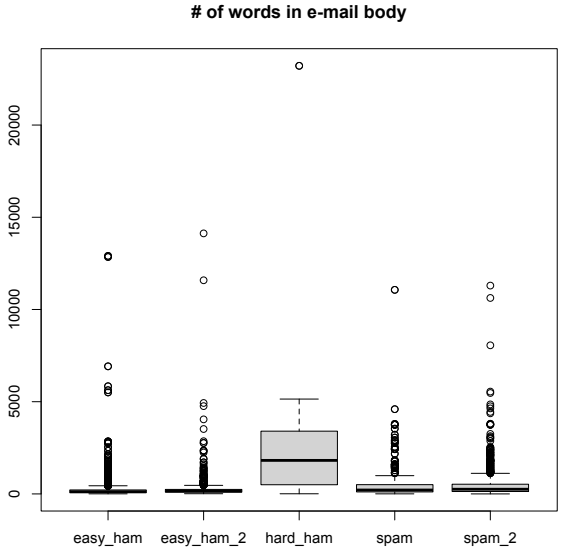
In order to make sure the algorithm is working correctly we must compare the left and right boundary tags. We manually copied the left boundary tags into a vector and compared them to the boundary from the extraction algorithm:

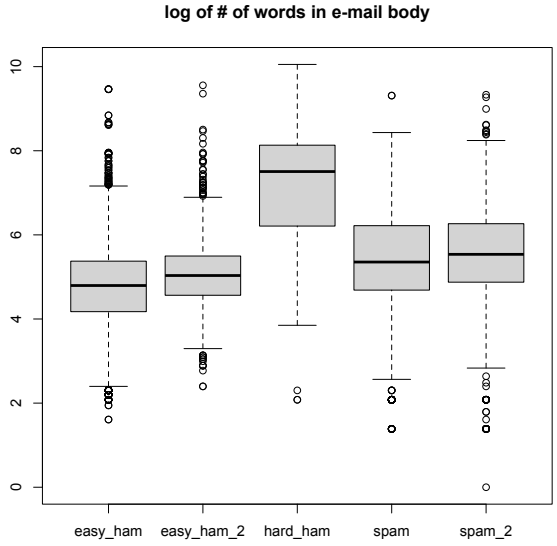
```
boundaries <- sapply(sampleHeaders[emailIndWithAttachments], getBoundary)
test <- c("==_Exmh_-1317289252P", "----=_NextPart_000_00C1_01C25017.F2F04E20", "Apple
-Mail-2-874629474", "_NextPart_1_bvfoDiTVghtoCXFdvJNKcuWblFV", "------------080808010
909060409040405", "==_Exmh_1547759024P", "----------=_1034083278-26594-4", "==_Exmh_-
1317289252P", "==_Exmh_-518574644P", "------------090602010909000705010009", "==_Exmh
_-403670396P", "==_Exmh_-398538836P", "==_Exmh_-763629846P", "==_Exmh_-451422450P")
test <- sapply(1:length(test), function(t) {
    return( test[1] == boundaries[1] )
})
```

```
> test
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

3. Number of words

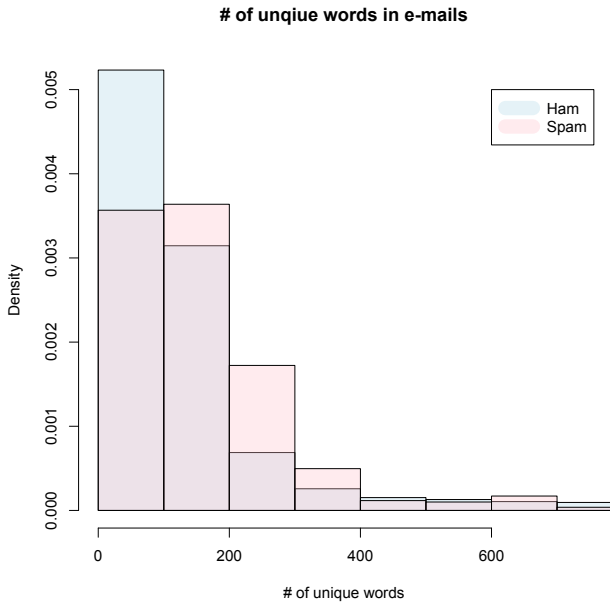
Spam Assassin has offered 5 different directories with e-mails.
The directories easy_ham, easy_ham_2 and hard_ham contain non-spam e-mail. The directories spam and spam_2 contain spam e-mail.
We isolated the e-mail body. For e-mails with attachments, we used the boundaries from the e-mail header as our two breakpoints. If the e-mail body only had one boundary in it, then everything after the boundary tag was the e-mail body.





The hard ham directory generally has the most words in the e-mail body. The other directories are all close together at almost equal word count. Spam Assassin designed hard_ham to also include some features of spam e-mail, which may have resulted in longer e-mail bodies.

4. Number of unique words



a. Unique words histogram

Looking at the previous boxplots, it does not appear like the number of words in the e-mail body can be used to predict whether or not the e-mail is spam.

We cut the number of unique words off at 750 in the histogram, since that encompasses the majority of the data points.

b. Unique words as a spam predictor

Spam e-mails do seem to have more unique words than ham e-mails. However, looking at the histogram, there are still a lot of ham e-mails in the buckets with more unique words. The bucket from 200-250 seems to be the most dominated by spam e-mails. But there are still over 1/3 as many ham e-mails in that bucket.

So, I do not think there is a clear cut-off to determine whether or not it is a spam e-mail. It can be said though that it is somewhat more likely to be a spam e-mail, if it has over 200 unique words.

5. Bayes Factor Derivation

In order to judge whether or not an e-mail is spam, we will calculate the Bayes Factor. When the Bayes Factor is greater than some C, we will make the assumption that the e-mail is spam. This is the derivation of the Bayes Factor:

Bayes Factor Derivation

We will derive the Bayes Factor, representing the evidence in favor of an e-mail being spam (relative to the evidence against it being spam.

Let $X = x_1, x_2, \ldots x_n$ be a feature vector of an e-mail, where every x corresponds to a word in our bag of words
For any x_i, if $x_i = 1$ then that word is in our e-mail.
If $x_i = 0$, then that word is not in our e-mail.

Let $P(spam|x)$ be the probability that the email is spam, given the features x.
Let $P(ham|x)$ be the probability that the email is ham, given the features x.
Let $P(spam)$ be the probability that an e-mail is spam.
Let $P(ham)$ be the probability that an e-mail is ham.
Let $P(x)$ be the probability that the features x are in an e-mail.

Bayes Law says that $P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$. So according to Bayes Law
$P(spam|x) = \frac{P(x|spam) * P(spam)}{P(x)}$ and $P(ham|x) = \frac{P(x|ham) * P(ham)}{P(x)}$.

Assume $x_1, x_2, \ldots, x_n$ are independant.
Therefore, $P(x|spam) = P(x_1|spam) \cdot P(x_2|spam) \cdot \ldots \cdot P(x_n|spam)$.
$P(x_n|spam)$ is the probability that an e-mail contains the word x_n, given that it's spam.

To calculate the probability of each word,
$P(x_i = 1 | spam) = \frac{\#\text{ of spam e-mails w/ } x_i + 0.1}{\#\text{ of spam emails} + 0.1}$
Add 0.1 to avoid errors, when spam e-mails do not contain word at all.

Once we calculate $P(spam|x)$ and $P(ham|x)$, we can calculate the Bayes Factor (BF):
$BF = \frac{\text{evidence supporting spam}}{\text{evidence against spam}} = \frac{P(spam|x)}{P(ham|x)}$
We can simplify it using Bayes Law,
$BF = \frac{P(x|spam) \cdot P(spam)}{P(x)} * \frac{P(x)}{P(x|ham) \cdot P(ham)} = \frac{P(x|spam)}{P(x|ham)} * \frac{P(spam)}{P(ham)}$

Assume that $P(spam) = P(ham) = 0.5$.
$BF = \frac{P(x|spam)}{P(x|ham} = \frac{P(x_1|spam) \cdot P(x_2|spam) \cdot \ldots \cdot P(x_n|spam)}{P(x_1|ham) \cdot P(x_2|ham) \cdot \ldots \cdot P(x_n|ham)}$

6. Probability of some word in spam vs. ham e-mails

As we can see from out Bayes Factor derivation, we will need to calculate the probability of seeing a word in spam and ham e-mails. We will do this just buy looking at how many spam/ham e-mails contain the word versus the total number of spam/ham e-mails.
Here are two examples that show the probabilities in action:

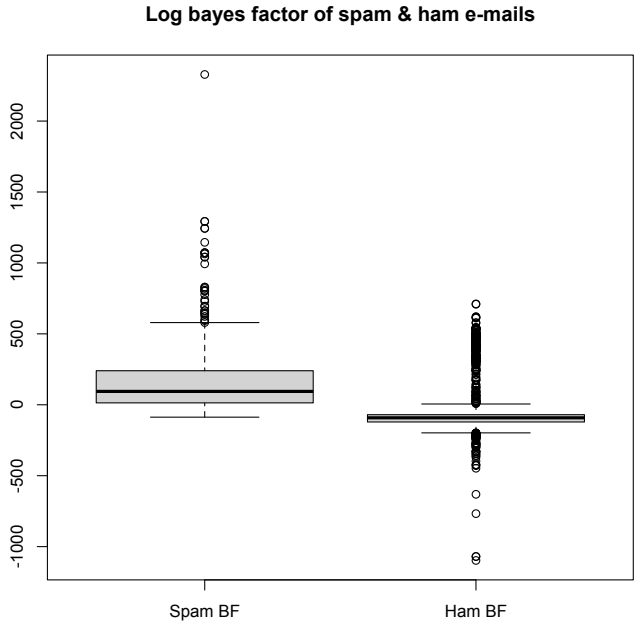
a. "monday"
It is 5.47495 more likely is it to see "monday" in a ham e-mail than a spam e-mail.

b. "buy"
It is 4.23894 times more likely to see "buy" in a spam e-mail than a ham e-mail.

7. Log Bayes Factor

a. Log
We will calculate the sum of the Bayes Factors on the log scale. When we take the log of an expression, multiplying translates to adding and dividing translates to subtracting. This avoids overflow values from complex multiplication. Thus, the log scale lets us keep our solution stable with such a large data set.

b. Boxplot
We split the Bayes Factors into spam & ham e-mails.



As we can see the Bayes Factors of the spam e-mails tend to be a lot higher than of the e-mails. This makes sense since the Bayes Factor measures the evidence in favor an e-mail being spam.

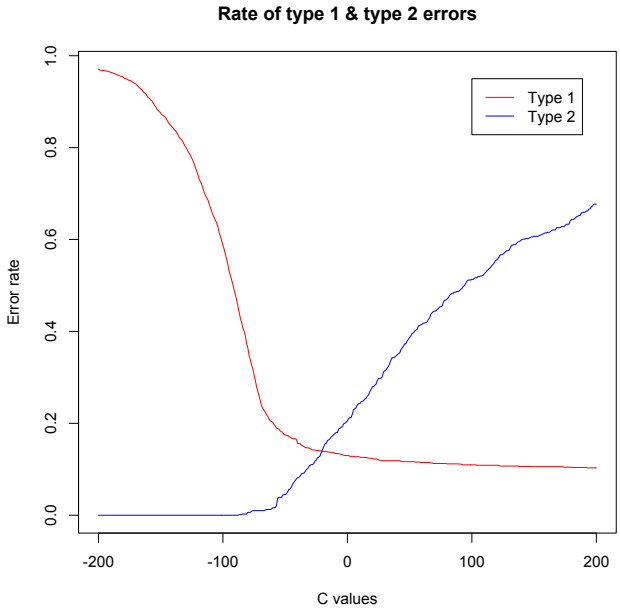
The IQR of the spam e-mails is much bigger than the IQR of ham e-mails. This makes sense, because the majority of ham e-mails will have a similar amount of evidence for them being spam: very little.

c. Cutoff "C" value

The value that we use to distinguish a spam and ham e-mail will be whether or not the Bayes Factor exceeds some constant C. Looking at the boxplot, the IQR of the spam e-mails seems to end right above 0. However, there are a lot of outliers in the ham e-mails that would be included, if we made cut-off at 0. Therefore, a good C estimate would be around 100.

8. Type-I & Type-II errors

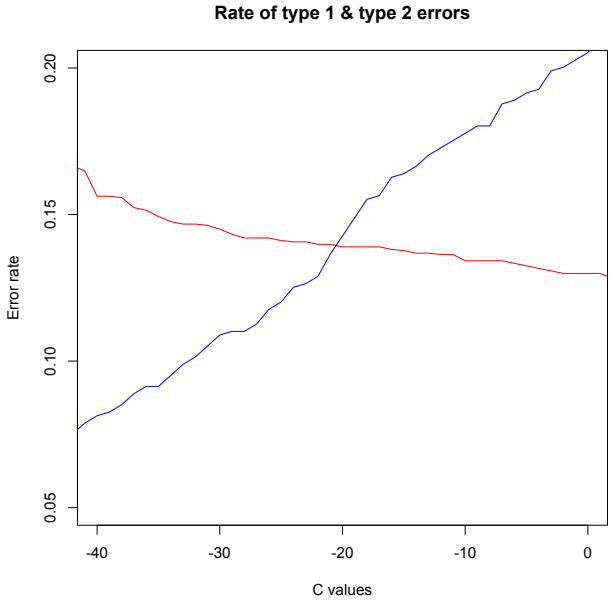
We will try to find the best c-value cutoff by observing the e-mails in our testing dataset.
A type 1 error occurs when we incorrectly categorize a ham e-mail as spam.
A type 2 error occurs when we incorrectly fail to categorize a spam e-mail as ham.



This plot shows the probability of type 1 and type 2 errors during the spam detection at different c values. We chose the range of v-values from -200 to 200, since that includes the majority of spam and ham e-mails.

In general, we can observe some trends about type 1 and type 2 errors as the c-value changes:
The lower the c-value, the higher the type 1 error rate and the lower the type 2 error rate.
The higher the c-value, the lower the type 2 error rate and the higher the type 1 error rate.

9. Same number of Type-I & Type-II errors



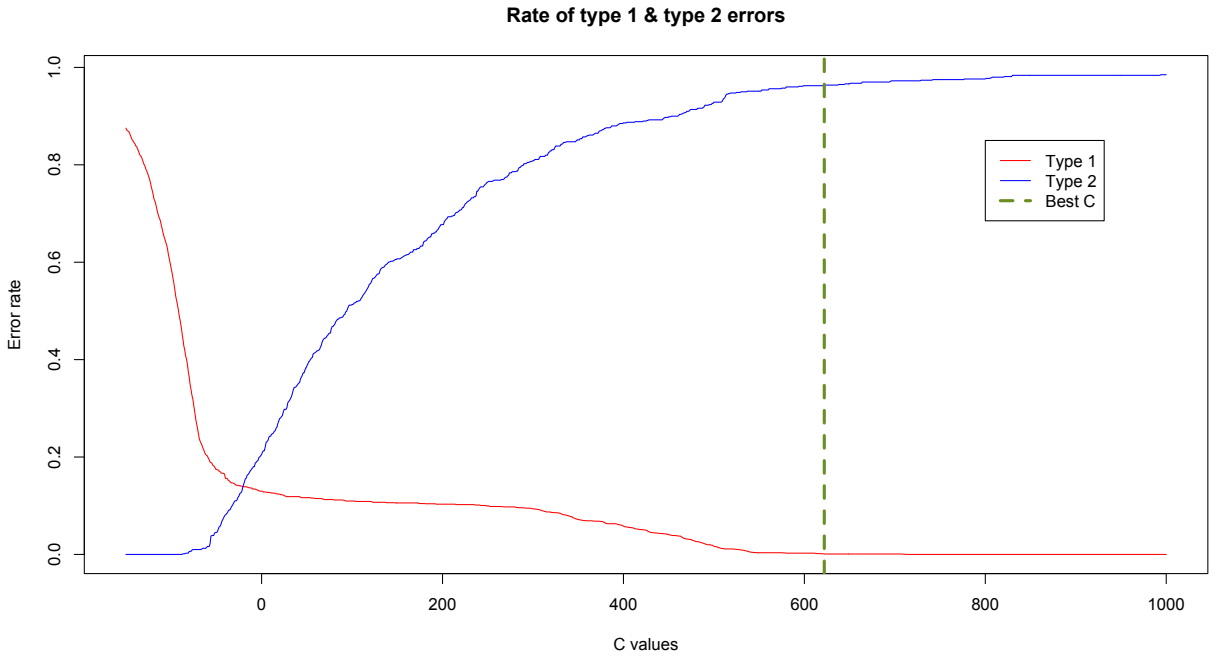
One way we could view the best C cut-off value is to find the C value that gives us the same number of type 1 and type 2 errors.

The C-value is at -21, where the type 1 error rate is 0.13983 and the type 2 error rate is 0.13642.

The graph to the left shows a zoom-in of our plot to the point where the type 1 and type2 error rates are the nearest.

10. Minimizing type-I errors

However, arguably the better way to find the best C cut-off value is to find the C value that gives us the least number of type 1 errors. The average user can most likely deal with the occasional spam e-mails. But if a very important e-mail is categorized as spam, this could significantly harm the user. In our case, it seems that type 1 errors are much more than type 2 errors.
We will find the C-value that makes the type 1 error rate <0.1%, while having the lowest type 2 error rate possible with that constraint.



In the plot above we see an extended version of the type 1 and type 2 error rate plot. Since, we want to minimize the type 1 error and type 1 error rate gets smaller the larger C gets, we extended the C values up to 1000. The green stripped line represents the point at which the type-1 error rate reaches below 0.1%. Since the type 2 error rate gets larger the larger C gets, we just need to find the first point that the type 1 error rate gets any lower than our threshold value of 0.1%.

The C value that gets the type 1 error rate below 0.1% with the smallest possible type 2 error rate is 622 with a type 1 error rate of 0.00086318 and a type 2 error rate of 0.96245.

This means that less than 0.1% of the ham e-mails will be categorized as spam. However, it also means that 96.2% of the spam e-mails will be failed to categorize as spam.
Overall, it's a trade-off, but it seems worth it to have a low type-1 error rate. The type 2 error rate starts reaching an asymptote around when C breaches 200. Beyond that point, we minimize our type 1 error rate without needing to grow our type 2 error rate too much.

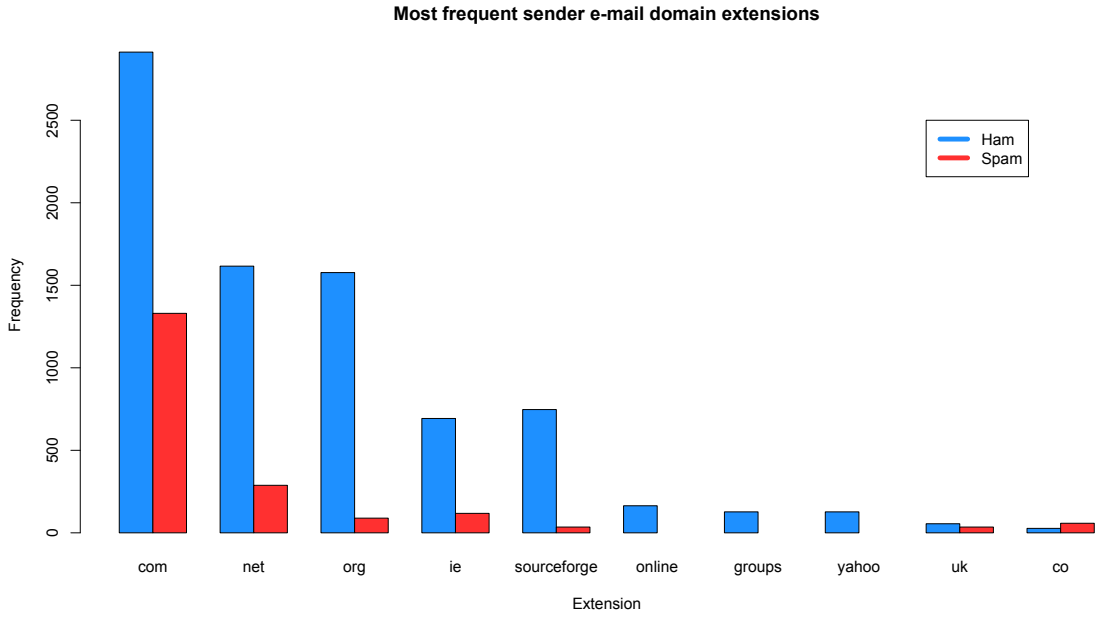
Extension

Throughout the majority of the project we've spent time looking at how we can detect a spam e-mail from the e-mail body. Now we will turn our attention to the e-mail header. However, the header contains a lot of data, so we will need to be selective about what we choose to focus on. Luckily, the first line of the header also contains the sender's e-mail address.

Using this this e-mail address, we will try to see if any of them can help us better understand what a spam e-mail looks like:
1. We will look at the domain extension of the e-mail address.
2. We will look at the length of, the number of unique character and the number of digits in the sender's username.

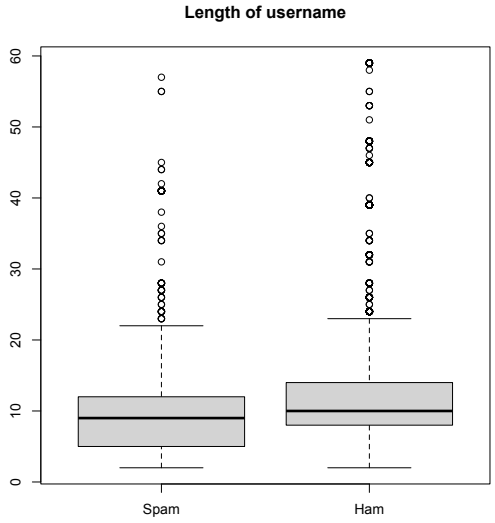
1. Domain extensions

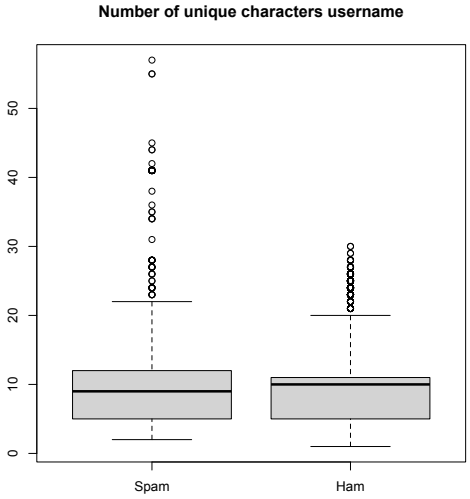
We extracted the e-mail domain extensions of the e-mail with the following criteria. Anything after the "@" and the first "." is in consideration for an extension. Then we split everything afterwards into a vector of strings on every ".".
There are several considerations we need to make when looking at the domain extensions of an e-mail address:
1. Many domains have multiple extensions. For instance, Stewart.Smith@ee.ed.ac.uk has three domain extensions: "ed", "ac", "uk". Therefore, we need to make sure that we extract all three of these.
2. It is important to realize how some website names are formatted. A "." does not always signify a website extension. It can be used for sub-websites, similarly to a "/".
For instance, "mail.google.com", has "mail", which is merely a precursor to the main domain of "google". Therefore, the way we extracted the domain extensions will actually also include several website names, instead of just extensions. We can handle this by not drawing any conclusions based on uncommon extensions. Instead we will only be looking at the most frequently used extensions.



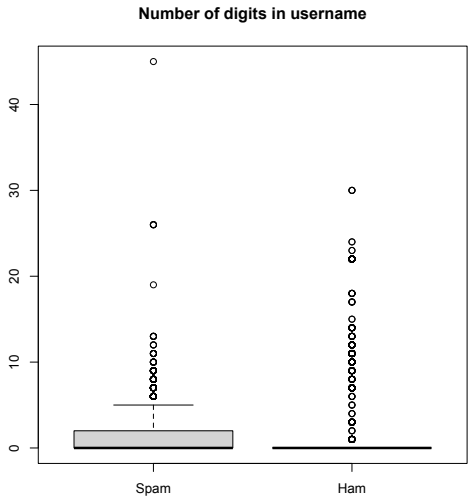
We made the bar plot on the previous page using the top 10 most frequently used in extensions in all of the e-mails. Then we split the extensions into ham, spam and extracted their frequency counts.
The most popular extension for both e-mail types is clearly ".com". However, the spam emails is dominated by ".com" to a greater degree: 55.48602 % of spam e-mails include a ".com" extension! On the other hand, the more eclectic domain extensions, such as ".online", ".groups", ".yahoo" have no spam e-mails. This is probably because they are more protected domain types. Therefore, we can draw the conclusion, that if a sender e-mail does not have a ".com" extension, it is more likely to be a ham e-mail.

2. Sender's username





Unfortunately, the sender's username does not provide a valuable indication of whether or not an e-mail is spam. The two boxplots above show two attempts to find some aspect of the username length that may indicate whether or not it is spam. However, I do not think either of them are successful at making a clear distinction between them.



The number of digits in the username has the biggest difference between ham and spam usernames from what we observed.
The mean number of digits in ham e-mails is 0.85138 (0 without outliers) and in spam e-mails is 1.36287 (0.65771 without outliers).
This difference is substantial. Therefore, we will draw the conclusion that if an e-mail includes at least some digits, it's more likely to be a spam e-mail.