DATA SCIENCE PROGRAMMING

LAB (L3+L4)

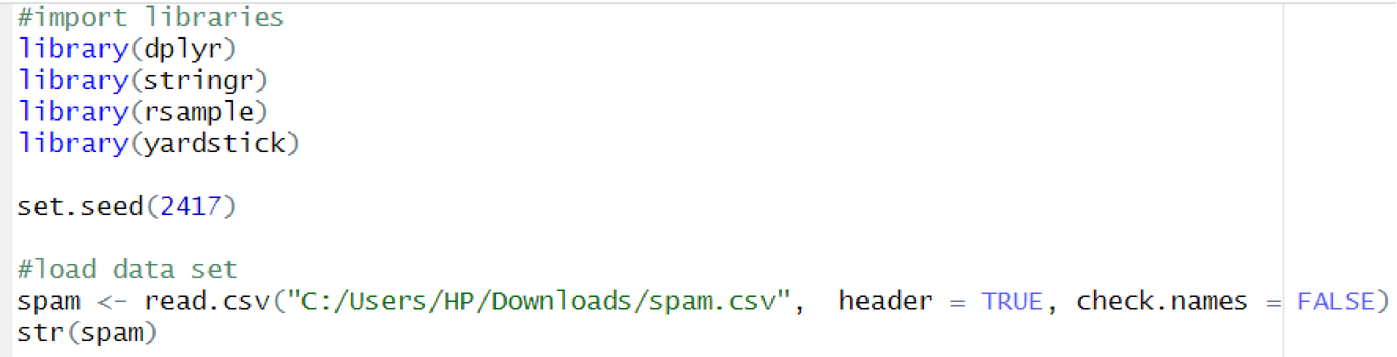
ASSESSMENT 2

NAÏVE BAYES IN R

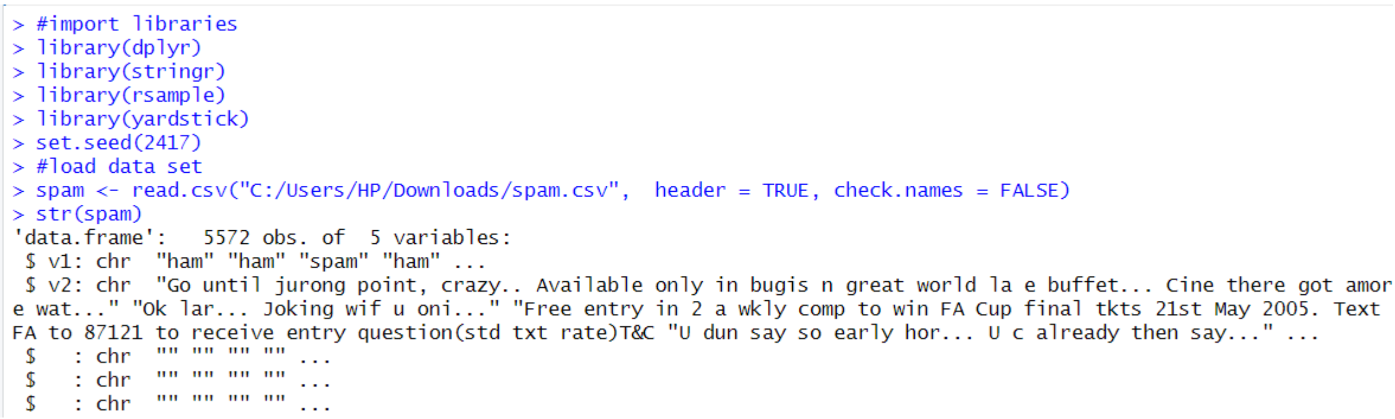
Name: SWATHI D

Reg. no.: 22MID0035

1. Import Libraries and Dataset

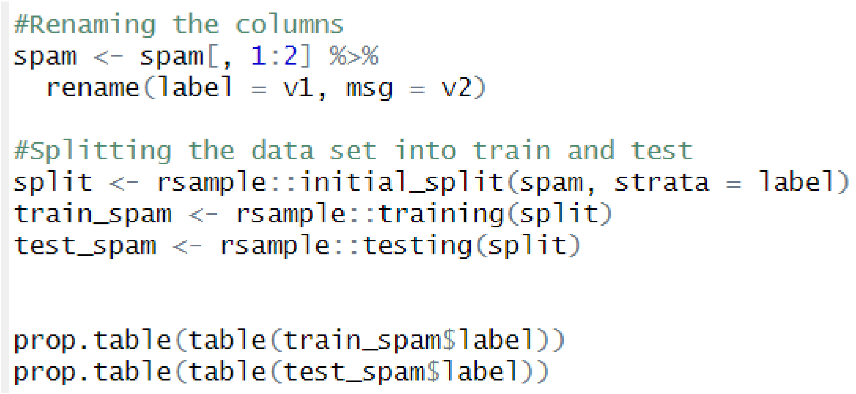


```
#import libraries
library(dplyr)
library(stringr)
library(rsample)
library(yardstick)

set.seed(2417)

#load data set
spam <- read.csv("C:/Users/HP/Downloads/spam.csv",  header = TRUE, check.names = FALSE)
str(spam)
```

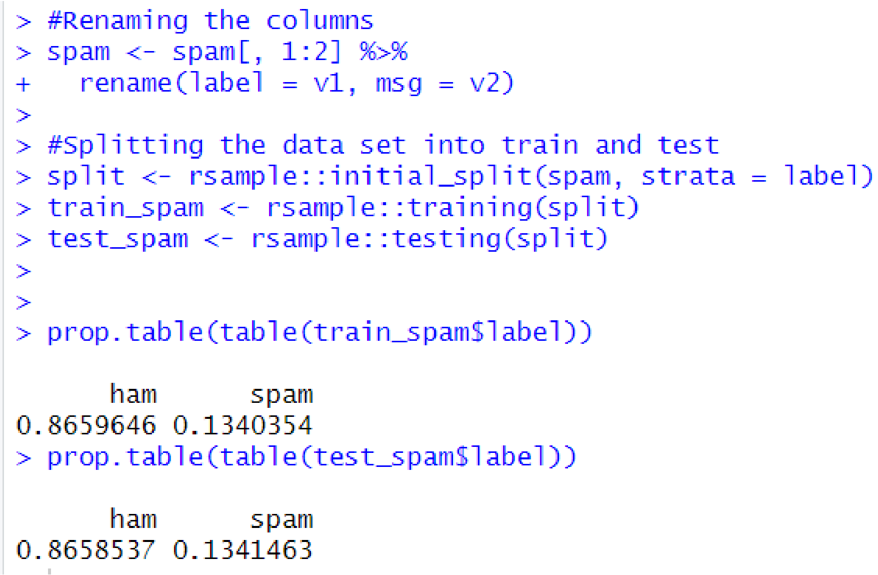


```
> #import libraries
> library(dplyr)
> library(stringr)
> library(rsample)
> library(yardstick)
> set.seed(2417)
> #load data set
> spam <- read.csv("C:/Users/HP/Downloads/spam.csv",  header = TRUE, check.names = FALSE)
> str(spam)
'data.frame':	5572 obs. of  5 variables:
 $ v1: chr  "ham" "ham" "spam" "ham" ...
 $ v2: chr  "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amor
e wat..." "Ok lar... Joking wif u oni..." "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text
FA to 87121 to receive entry question(std txt rate)T&C "U dun say so early hor... U c already then say..." ...
 $   : chr  "" "" "" "" ...
 $   : chr  "" "" "" "" ...
 $   : chr  "" "" "" "" ...
```

2. Read and prepare the dataset

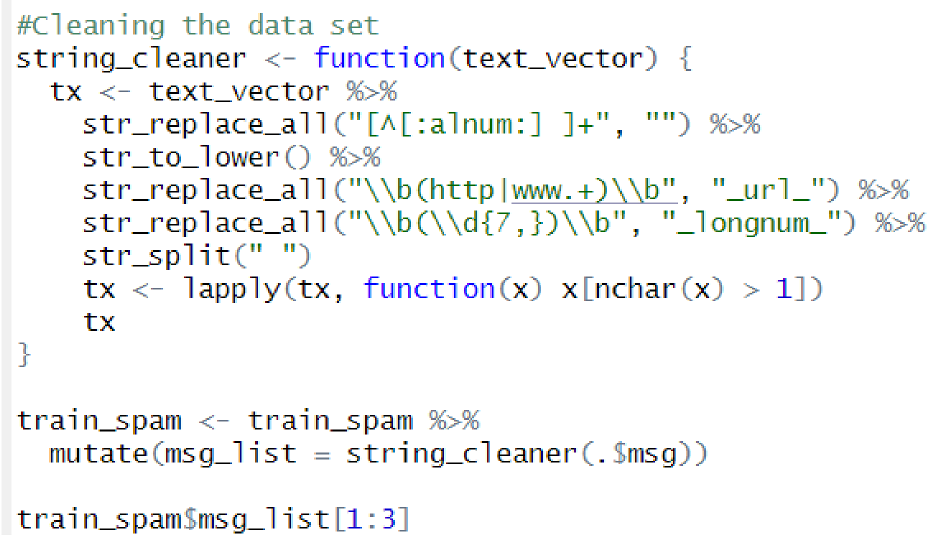


```r
#Renaming the columns
spam <- spam[, 1:2] %>%
  rename(label = v1, msg = v2)

#Splitting the data set into train and test
split <- rsample::initial_split(spam, strata = label)
train_spam <- rsample::training(split)
test_spam <- rsample::testing(split)


prop.table(table(train_spam$label))
prop.table(table(test_spam$label))
```

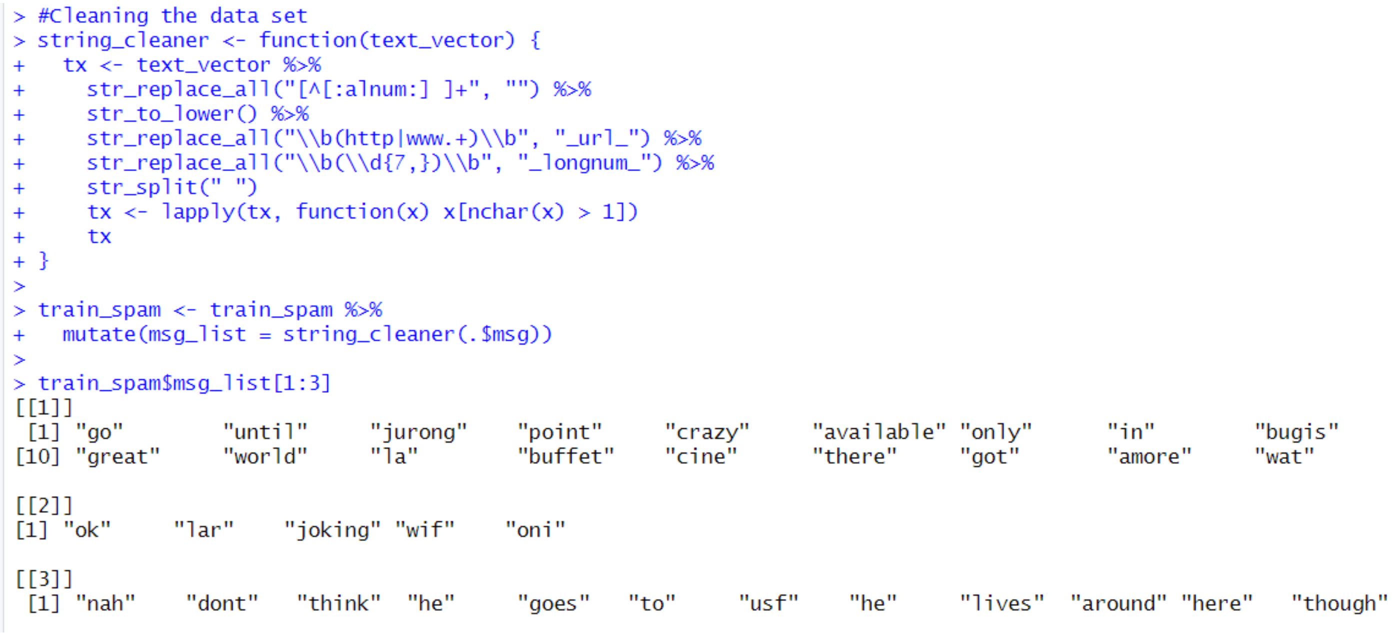


```
> #Renaming the columns
> spam <- spam[, 1:2] %>%
+   rename(label = v1, msg = v2)
>
> #Splitting the data set into train and test
> split <- rsample::initial_split(spam, strata = label)
> train_spam <- rsample::training(split)
> test_spam <- rsample::testing(split)
>
>
> prop.table(table(train_spam$label))

      ham      spam
0.8659646 0.1340354
> prop.table(table(test_spam$label))

      ham      spam
0.8658537 0.1341463
```

3. Cleaning the dataset

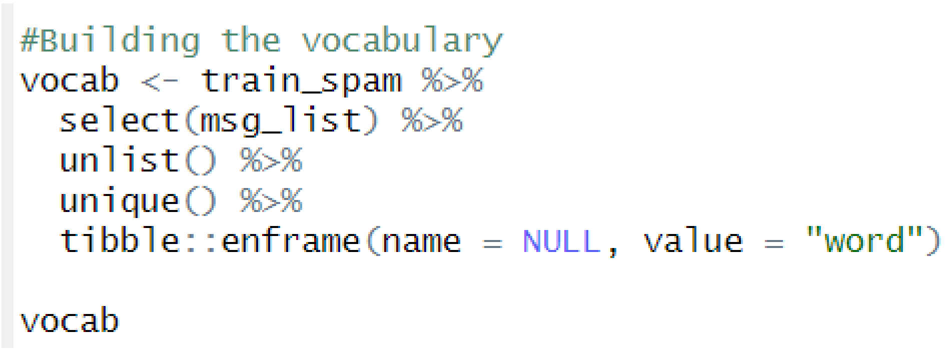


```r
#Cleaning the data set
string_cleaner <- function(text_vector) {
  tx <- text_vector %>%
    str_replace_all("[^[:alnum:] ]+", "") %>%
    str_to_lower() %>%
    str_replace_all("\\b(http|www.+)\\b", "_url_") %>%
    str_replace_all("\\b(\\d{7,})\\b", "_longnum_") %>%
    str_split(" ")
    tx <- lapply(tx, function(x) x[nchar(x) > 1])
    tx
}

train_spam <- train_spam %>%
  mutate(msg_list = string_cleaner(.$msg))

train_spam$msg_list[1:3]
```

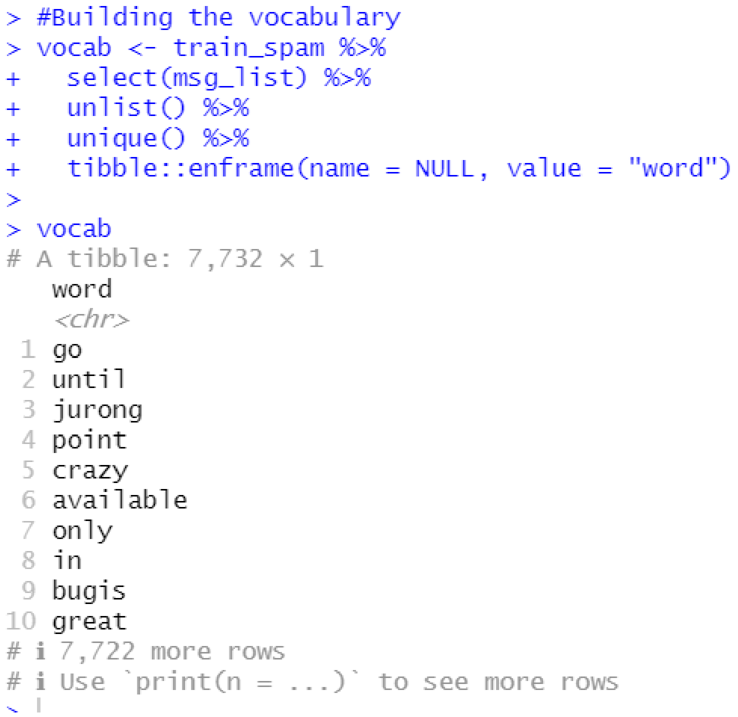


```
> #Cleaning the data set
> string_cleaner <- function(text_vector) {
+   tx <- text_vector %>%
+     str_replace_all("[^[:alnum:] ]+", "") %>%
+     str_to_lower() %>%
+     str_replace_all("\\b(http|www.+)\\b", "_url_") %>%
+     str_replace_all("\\b(\\d{7,})\\b", "_longnum_") %>%
+     str_split(" ")
+     tx <- lapply(tx, function(x) x[nchar(x) > 1])
+     tx
+ }
>
> train_spam <- train_spam %>%
+   mutate(msg_list = string_cleaner(.$msg))
>
> train_spam$msg_list[1:3]
[[1]]
 [1] "go"        "until"     "jurong"    "point"     "crazy"     "available" "only"      "in"        "bugis"
[10] "great"     "world"     "la"        "buffet"    "cine"      "there"     "got"       "amore"     "wat"

[[2]]
[1] "ok"     "lar"    "joking" "wif"    "oni"

[[3]]
 [1] "nah"    "dont"   "think"  "he"     "goes"   "to"     "usf"    "he"     "lives"  "around" "here"   "though"
```

4. Building the vocabulary

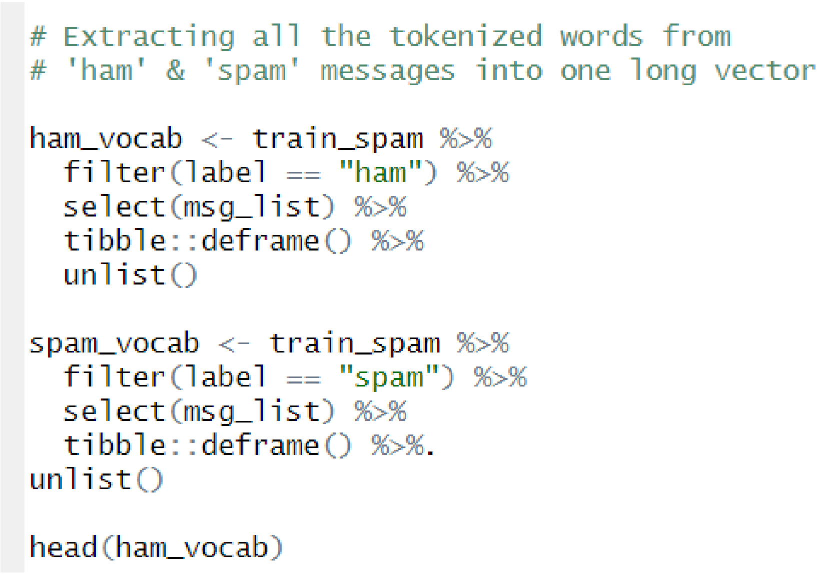


```
#Building the vocabulary
vocab <- train_spam %>%
  select(msg_list) %>%
  unlist() %>%
  unique() %>%
  tibble::enframe(name = NULL, value = "word")

vocab
```

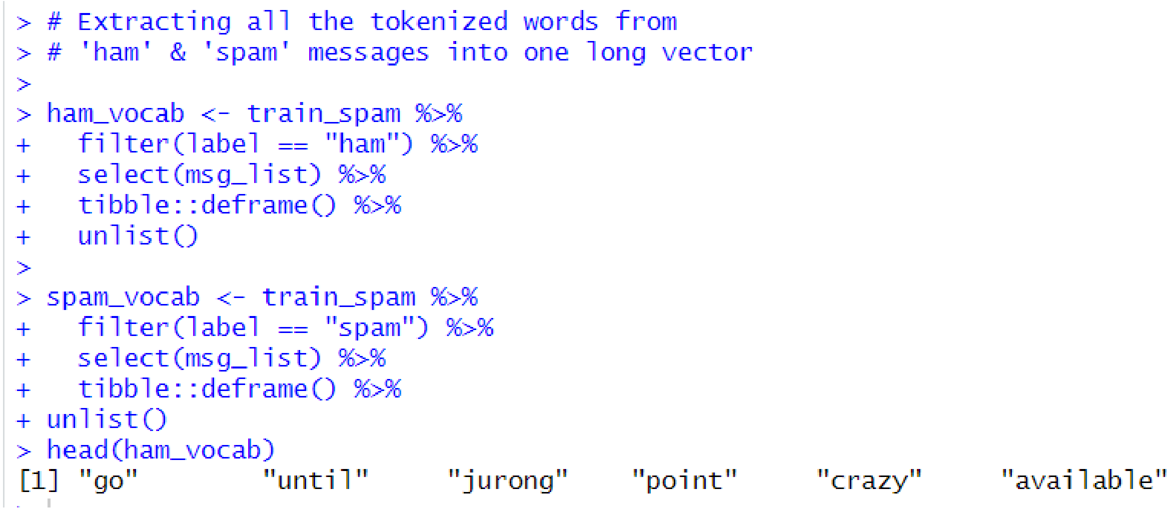


```
> #Building the vocabulary
> vocab <- train_spam %>%
+   select(msg_list) %>%
+   unlist() %>%
+   unique() %>%
+   tibble::enframe(name = NULL, value = "word")
>
> vocab
# A tibble: 7,732 × 1
   word
   <chr>
 1 go
 2 until
 3 jurong
 4 point
 5 crazy
 6 available
 7 only
 8 in
 9 bugis
10 great
# ℹ 7,722 more rows
# ℹ Use `print(n = ...)` to see more rows
>
```

5. Separate ham and spam vocab

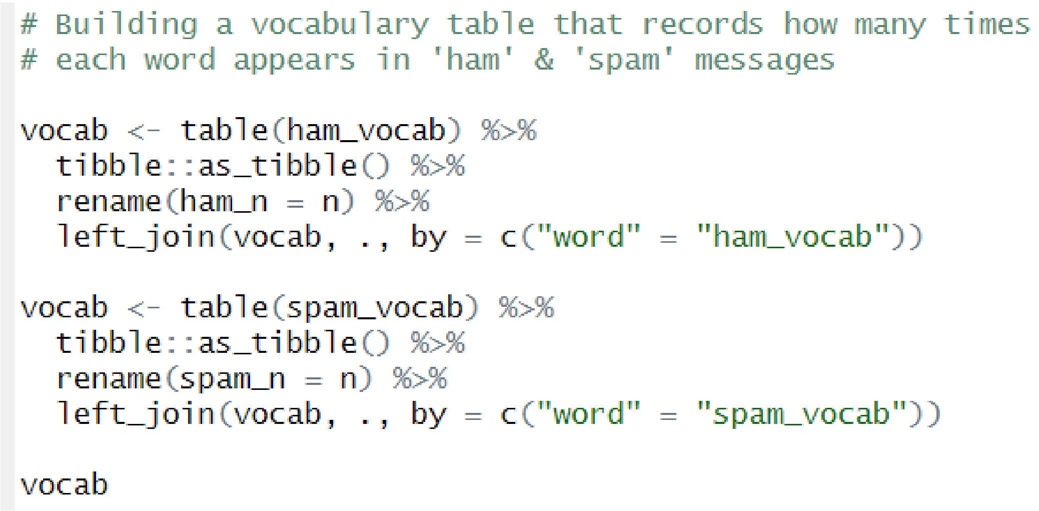


```
# Extracting all the tokenized words from
# 'ham' & 'spam' messages into one long vector

ham_vocab <- train_spam %>%
  filter(label == "ham") %>%
  select(msg_list) %>%
  tibble::deframe() %>%
  unlist()

spam_vocab <- train_spam %>%
  filter(label == "spam") %>%
  select(msg_list) %>%
  tibble::deframe() %>%.
unlist()

head(ham_vocab)
```

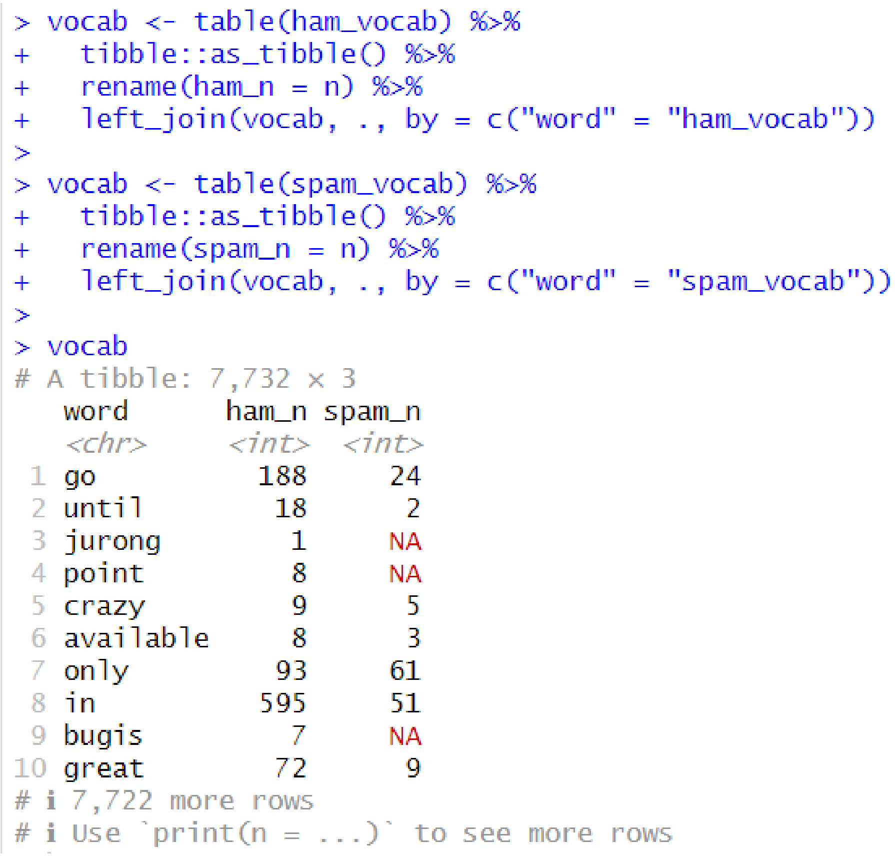


```
> # Extracting all the tokenized words from
> # 'ham' & 'spam' messages into one long vector
>
> ham_vocab <- train_spam %>%
+   filter(label == "ham") %>%
+   select(msg_list) %>%
+   tibble::deframe() %>%
+   unlist()
>
> spam_vocab <- train_spam %>%
+   filter(label == "spam") %>%
+   select(msg_list) %>%
+   tibble::deframe() %>%
+ unlist()
> head(ham_vocab)
[1] "go"        "until"     "jurong"    "point"     "crazy"     "available"
```

6. Count word frequencies

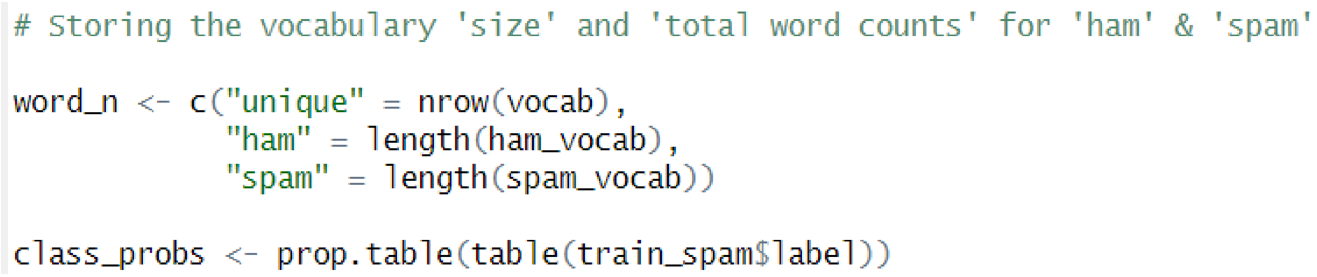


```
# Building a vocabulary table that records how many times
# each word appears in 'ham' & 'spam' messages

vocab <- table(ham_vocab) %>%
  tibble::as_tibble() %>%
  rename(ham_n = n) %>%
  left_join(vocab, ., by = c("word" = "ham_vocab"))

vocab <- table(spam_vocab) %>%
  tibble::as_tibble() %>%
  rename(spam_n = n) %>%
  left_join(vocab, ., by = c("word" = "spam_vocab"))

vocab
```

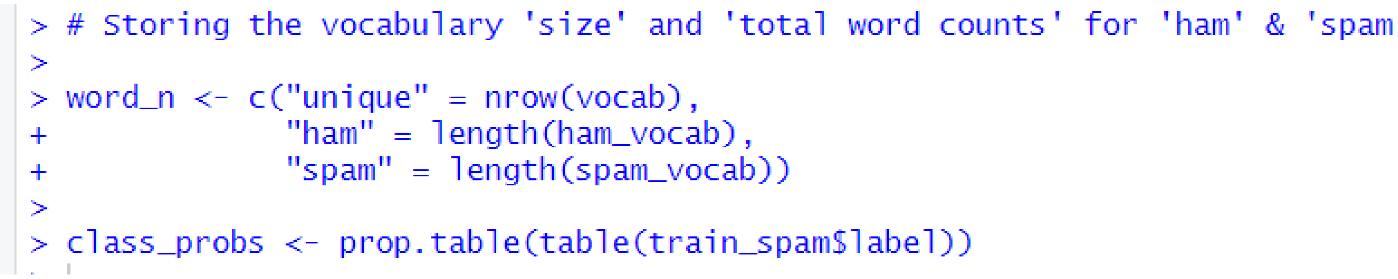


```
> vocab <- table(ham_vocab) %>%
+   tibble::as_tibble() %>%
+   rename(ham_n = n) %>%
+   left_join(vocab, ., by = c("word" = "ham_vocab"))
>
> vocab <- table(spam_vocab) %>%
+   tibble::as_tibble() %>%
+   rename(spam_n = n) %>%
+   left_join(vocab, ., by = c("word" = "spam_vocab"))
>
> vocab
# A tibble: 7,732 × 3
   word      ham_n spam_n
   <chr>     <int>  <int>
 1 go          188     24
 2 until        18      2
 3 jurong        1     NA
 4 point         8     NA
 5 crazy         9      5
 6 available     8      3
 7 only         93     61
 8 in          595     51
 9 bugis         7     NA
10 great        72      9
# ℹ 7,722 more rows
# ℹ Use `print(n = ...)` to see more rows
```

7. Store totals

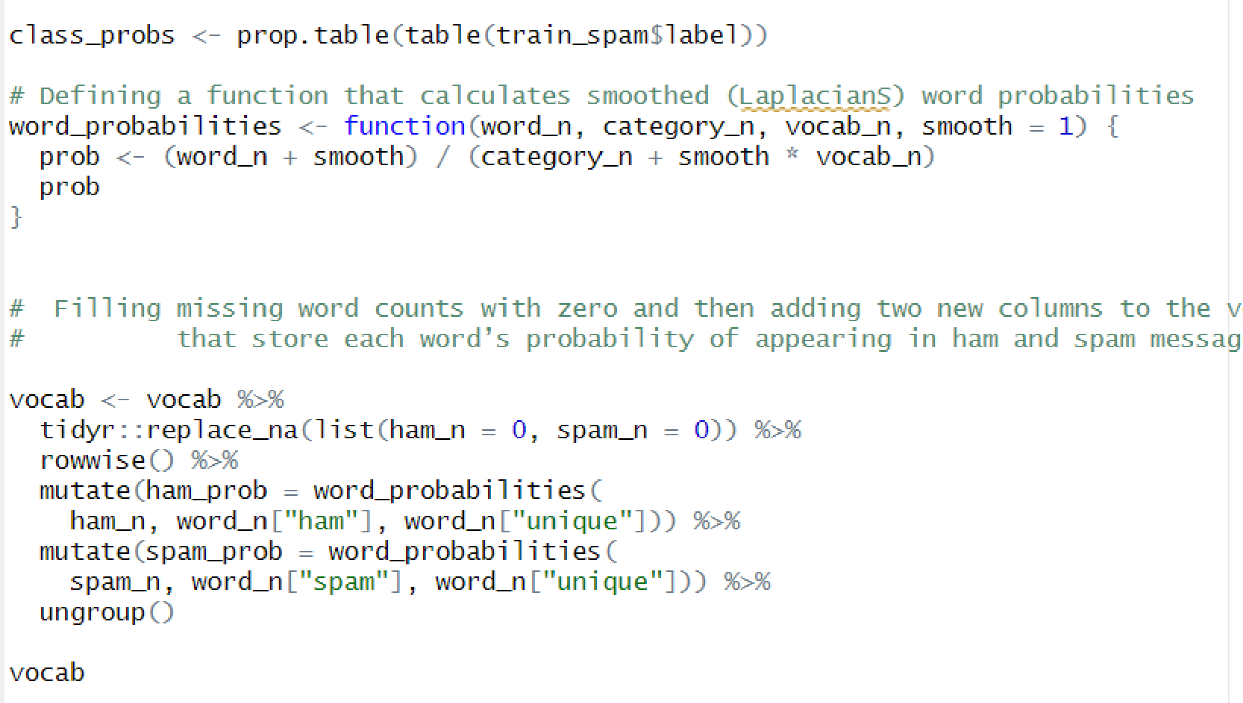


```
# Storing the vocabulary 'size' and 'total word counts' for 'ham' & 'spam'

word_n <- c("unique" = nrow(vocab),
            "ham" = length(ham_vocab),
            "spam" = length(spam_vocab))

class_probs <- prop.table(table(train_spam$label))
```

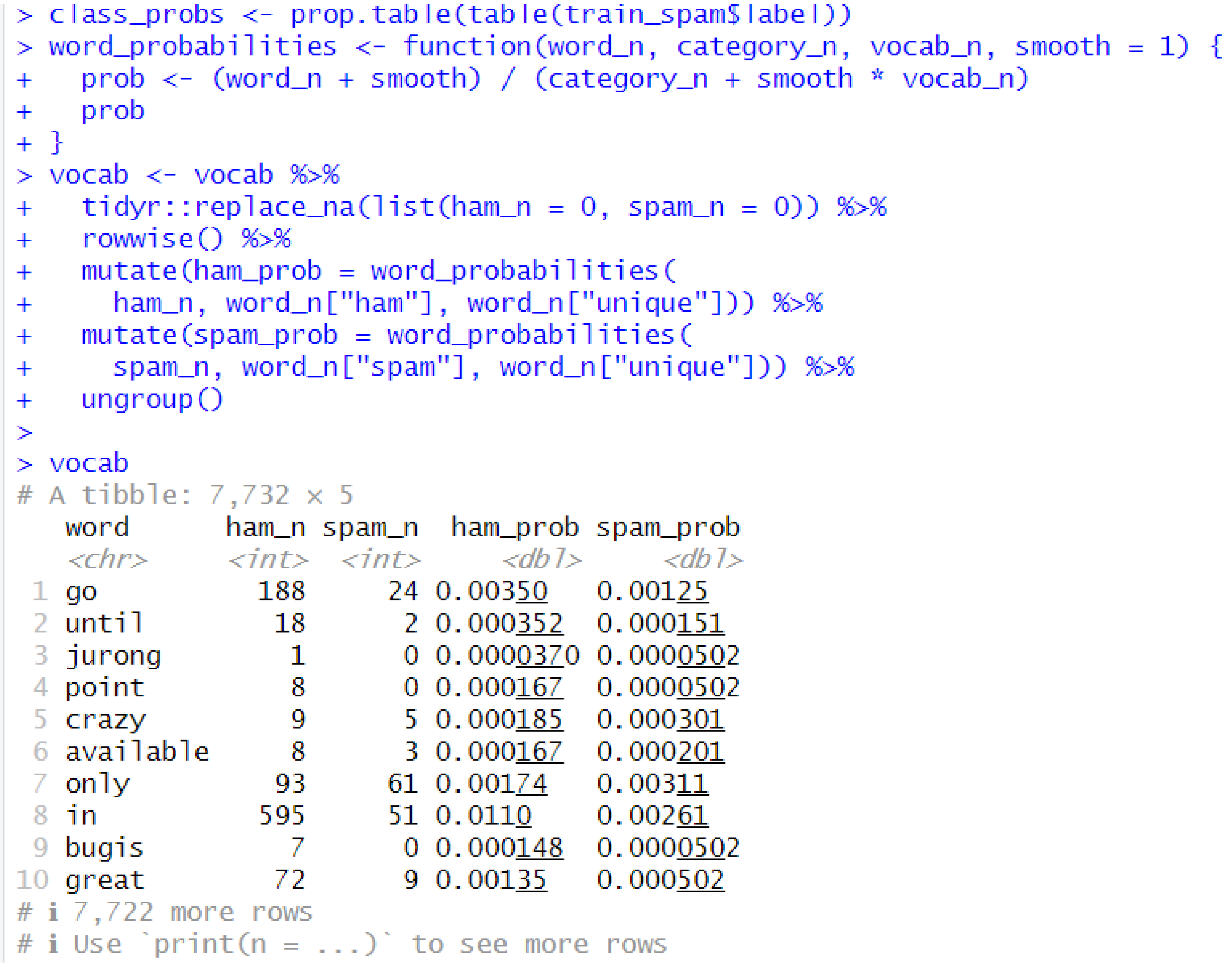


```
> # Storing the vocabulary 'size' and 'total word counts' for 'ham' & 'spam
>
> word_n <- c("unique" = nrow(vocab),
+             "ham" = length(ham_vocab),
+             "spam" = length(spam_vocab))
>
> class_probs <- prop.table(table(train_spam$label))
```

8. Define word probability function

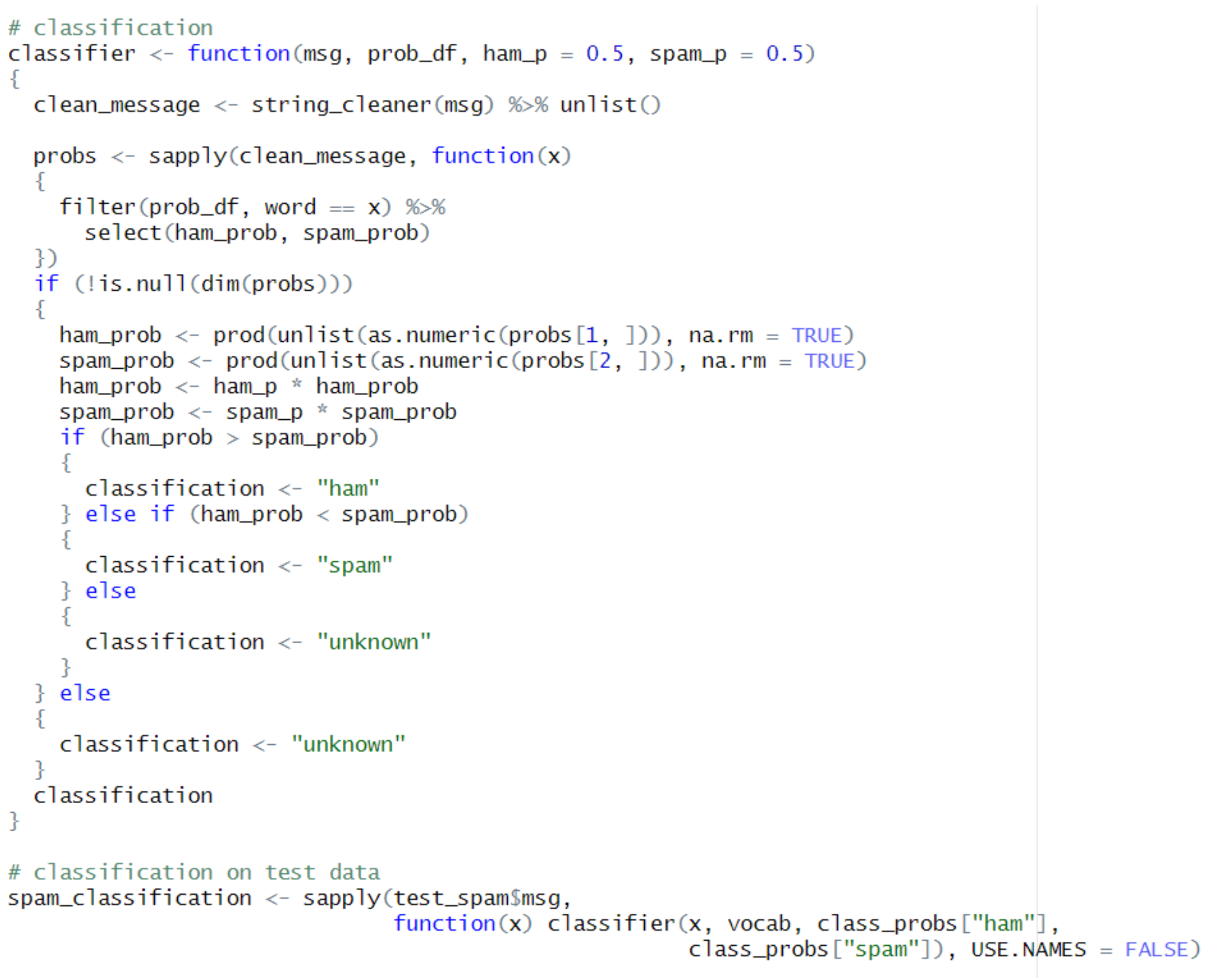


```
class_probs <- prop.table(table(train_spam$label))

# Defining a function that calculates smoothed (LaplacianS) word probabilities
word_probabilities <- function(word_n, category_n, vocab_n, smooth = 1) {
  prob <- (word_n + smooth) / (category_n + smooth * vocab_n)
  prob
}


#  Filling missing word counts with zero and then adding two new columns to the vocabulary
#          that store each word’s probability of appearing in ham and spam messages

vocab <- vocab %>%
  tidyr::replace_na(list(ham_n = 0, spam_n = 0)) %>%
  rowwise() %>%
  mutate(ham_prob = word_probabilities(
    ham_n, word_n["ham"], word_n["unique"])) %>%
  mutate(spam_prob = word_probabilities(
    spam_n, word_n["spam"], word_n["unique"])) %>%
  ungroup()

vocab
```

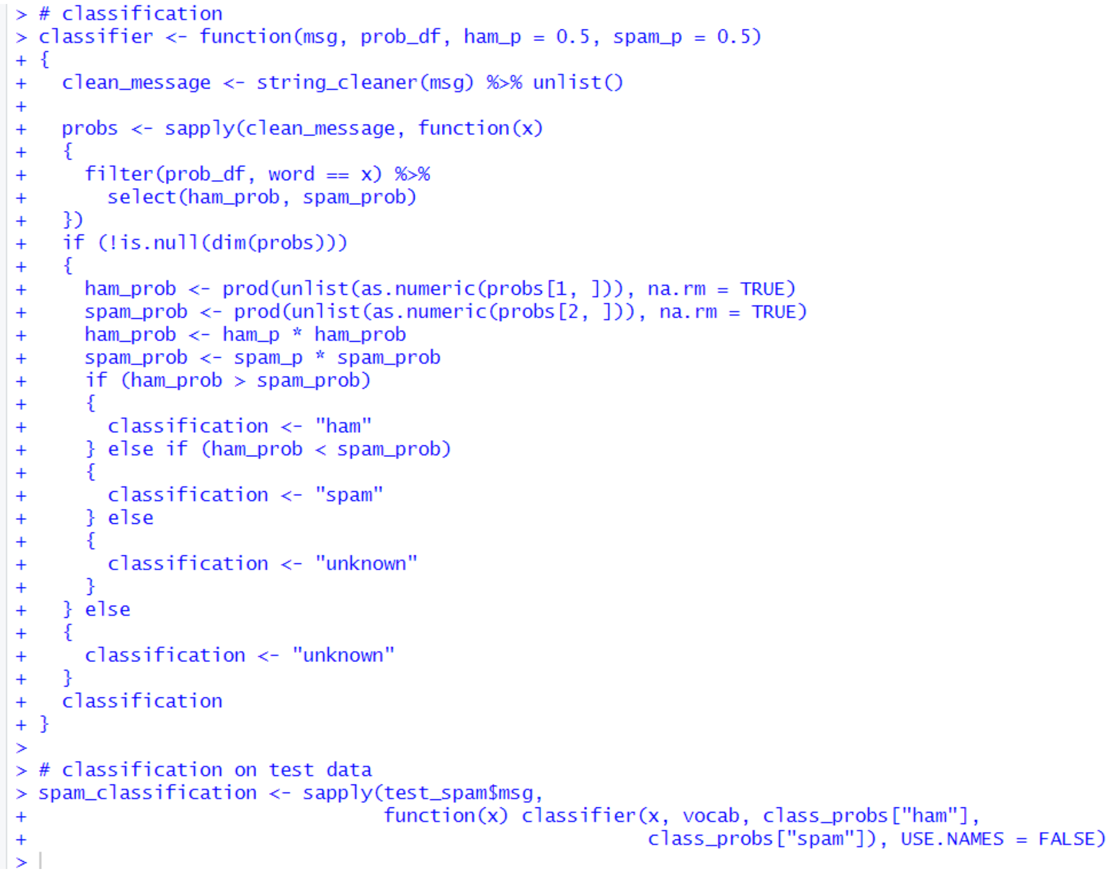


```
> class_probs <- prop.table(table(train_spam$label))
> word_probabilities <- function(word_n, category_n, vocab_n, smooth = 1) {
+   prob <- (word_n + smooth) / (category_n + smooth * vocab_n)
+   prob
+ }
> vocab <- vocab %>%
+   tidyr::replace_na(list(ham_n = 0, spam_n = 0)) %>%
+   rowwise() %>%
+   mutate(ham_prob = word_probabilities(
+     ham_n, word_n["ham"], word_n["unique"])) %>%
+   mutate(spam_prob = word_probabilities(
+     spam_n, word_n["spam"], word_n["unique"])) %>%
+   ungroup()
> 
> vocab
# A tibble: 7,732 × 5
   word      ham_n spam_n  ham_prob spam_prob
   <chr>     <int>  <int>     <dbl>     <dbl>
 1 go          188     24 0.00350   0.00125  
 2 until        18      2 0.000352  0.000151 
 3 jurong        1      0 0.0000370 0.0000502
 4 point         8      0 0.000167  0.0000502
 5 crazy         9      5 0.000185  0.000301 
 6 available     8      3 0.000167  0.000201 
 7 only         93     61 0.00174   0.00311  
 8 in          595     51 0.0110    0.00261  
 9 bugis         7      0 0.000148  0.0000502
10 great        72      9 0.00135   0.000502 
# ℹ 7,722 more rows
# ℹ Use `print(n = ...)` to see more rows
```

9. Define classifier function

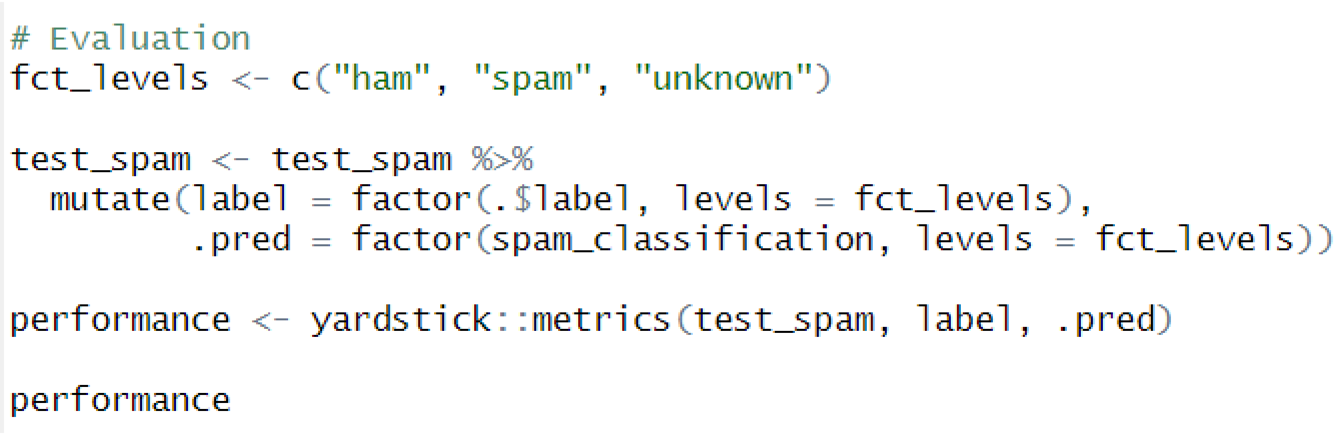


```
# classification
classifier <- function(msg, prob_df, ham_p = 0.5, spam_p = 0.5)
{
  clean_message <- string_cleaner(msg) %>% unlist()

  probs <- sapply(clean_message, function(x)
  {
    filter(prob_df, word == x) %>%
      select(ham_prob, spam_prob)
  })
  if (!is.null(dim(probs)))
  {
    ham_prob <- prod(unlist(as.numeric(probs[1, ])), na.rm = TRUE)
    spam_prob <- prod(unlist(as.numeric(probs[2, ])), na.rm = TRUE)
    ham_prob <- ham_p * ham_prob
    spam_prob <- spam_p * spam_prob
    if (ham_prob > spam_prob)
    {
      classification <- "ham"
    } else if (ham_prob < spam_prob)
    {
      classification <- "spam"
    } else
    {
      classification <- "unknown"
    }
  } else
  {
    classification <- "unknown"
  }
  classification
}

# classification on test data
spam_classification <- sapply(test_spam$msg,
                              function(x) classifier(x, vocab, class_probs["ham"],
                                                     class_probs["spam"]), USE.NAMES = FALSE)
```

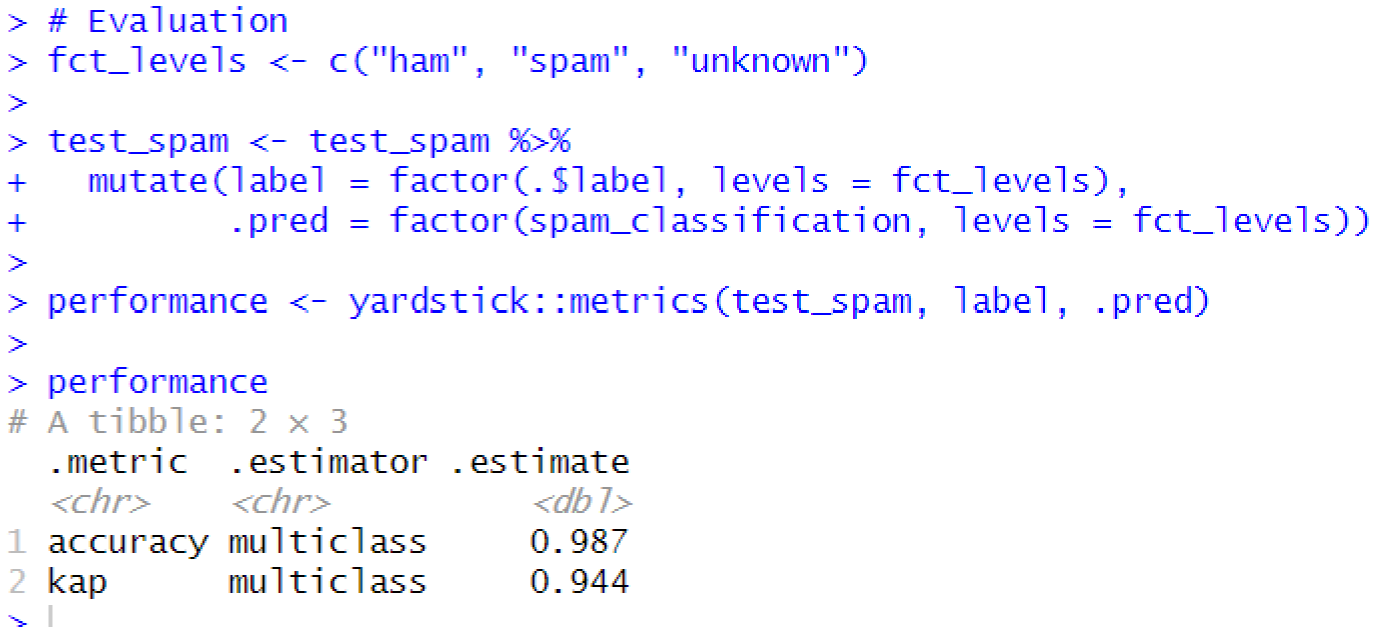


```
> # classification
> classifier <- function(msg, prob_df, ham_p = 0.5, spam_p = 0.5)
+ {
+   clean_message <- string_cleaner(msg) %>% unlist()
+ 
+   probs <- sapply(clean_message, function(x)
+   {
+     filter(prob_df, word == x) %>%
+       select(ham_prob, spam_prob)
+   })
+   if (!is.null(dim(probs)))
+   {
+     ham_prob <- prod(unlist(as.numeric(probs[1, ])), na.rm = TRUE)
+     spam_prob <- prod(unlist(as.numeric(probs[2, ])), na.rm = TRUE)
+     ham_prob <- ham_p * ham_prob
+     spam_prob <- spam_p * spam_prob
+     if (ham_prob > spam_prob)
+     {
+       classification <- "ham"
+     } else if (ham_prob < spam_prob)
+     {
+       classification <- "spam"
+     } else
+     {
+       classification <- "unknown"
+     }
+   } else
+   {
+     classification <- "unknown"
+   }
+   classification
+ }
> 
> # classification on test data
> spam_classification <- sapply(test_spam$msg,
+                               function(x) classifier(x, vocab, class_probs["ham"],
+                                                      class_probs["spam"]), USE.NAMES = FALSE)
> 
```

10. Evaluation



```
# Evaluation
fct_levels <- c("ham", "spam", "unknown")

test_spam <- test_spam %>%
  mutate(label = factor(.$label, levels = fct_levels),
         .pred = factor(spam_classification, levels = fct_levels))

performance <- yardstick::metrics(test_spam, label, .pred)

performance
```



```
> # Evaluation
> fct_levels <- c("ham", "spam", "unknown")
> 
> test_spam <- test_spam %>%
+   mutate(label = factor(.$label, levels = fct_levels),
+          .pred = factor(spam_classification, levels = fct_levels))
> 
> performance <- yardstick::metrics(test_spam, label, .pred)
> 
> performance
# A tibble: 2 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy multiclass     0.987
2 kap      multiclass     0.944
> 
```