STAT 33A Homework 7

CJ HINES (3034590053)

Dec 10, 2020

This homework is due **Dec 10, 2020** by 11:59pm PT.

Homeworks are graded for correctness.

As you work, write your answers in this notebook. Answer questions with complete sentences, and put code in code chunks. You can make as many new code chunks as you like.

Please do not delete the exercises already in this notebook, because it may interfere with our grading tools.

You need to submit your work in two places:

- Submit this Rmd file with your edits on bCourses.
- Knit and submit the generated PDF file on Gradescope.

If you have any last-minute trouble knitting, **DON'T PANIC**. Submit your Rmd file on time and follow up in office hours or on Piazza to sort out the PDF.

SpamAssassin Email Data

The SpamAssassin Email Data set is a collection of email messages used to train the SpamAssassin software to detect spam. The email messages are divided into legitimate "ham" emails and illegitimate "spam" emails. Each email is in a separate plain text file.

In this assignment, you'll only use a collection of "ham" emails. You can find the emails in the file `emails.zip` on the bCourse. You will need to unzip the file before proceeding with Exercise 1.

This data set is originally from the Apache SpamAssassin project.

Exercise 1

The `readLines` function reads lines of text from a file and returns them in a character vector with one element for each line. The first argument is the path to the file. By default, the function will read all of the lines in the file.

Write a function `read_email` that reads all of the text in a single email file. Your function should have a parameter `file` to set the path to the file. Your function should collapse all of the lines in the file into a single string with lines separated by the newline character `\n`.

Show that your function works for 3 of the email files.

Hint: The `paste` function is relevant here.

YOUR ANSWER GOES HERE:

```r
read_email = function(file) {
  paste(readLines(file), collapse = "\n")
}

#Tests
t1 = read_email("easy_ham/0001.ea7e79d3153e7469e7a9c3e0af6a357e")
t2 = read_email("easy_ham/0002.b3120c4bcbf3101e661161ee7efcb8bf")
t3 = read_email("easy_ham/0559.2c60829e4147cdc981cf7131088b851d")
head(t1)
```

```
## [1] "From exmh-workers-admin@redhat.com  Thu Aug 22 12:36:23 2002\nReturn-Path: <exmh-workers-admin@
```

Exercise 2

Write a function `read_email_all` that reads all of the files in the email directory and returns a character vector with one element for each email. Your function should have a parameter `dir` to set the path to the directory. Your function should call the `read_email` function from Exercise 1.

Make sure not to put other files in the email directory!

After writing your function, use it to read all of the email files into a character vector called `emails`.

How many email files are there?

Hint: The `list.files` function is relevant here.

YOUR ANSWER GOES HERE:

```r
read_email_all = function(dir) {
  emails = character(length(list.files(dir())))
  sapply(list.files(dir, full.names = TRUE), read_email)
}

emails = read_email_all("easy_ham")
length(emails)
```

```
## [1] 2551
```

> There are 2551 email files.

Exercise 3

Use stringr and regular expressions to write a function `extract_email_addr` that extracts all email addresses from a character vector. Your function should have a parameter `x` for the character vector, and should return a character vector with one element for each email address (duplicates are okay).

For simplicity, you can assume the formatting rules for email addresses are that they:

1. Must contain exactly one at-symbol `@`
2. Can also contain any number of letters, numbers, or characters in `._-`

Test your function on some made up strings and also on one of the email messages.

Hint: Using character classes [ ] in the regex pattern is important here.

Note: It's not necessary for this exercise, but if you're curious about the actual rules for email addresses, see Section 3.4.1 of RFC 5322, or this Wikipedia article.

YOUR ANSWER GOES HERE:

```r
library(stringr)

extract_email_addr = function(x) {
  addresses = str_extract_all(x, "[[:alnum:]]+@[[:alpha:]]+\\.[[:alpha:]]+")
  return(addresses)
}

#Tests
realtest = extract_email_addr(emails[5])
realtest
```

```
## [[1]]
##  [1] "admin@redhat.com"                "admin@example.com"
##  [3] "zzzz@localhost.netnoteinc"       "exmh@example.com"
##  [5] "users@listman.example"           "users@listman.redhat"
##  [7] "users@listman.redhat"            "users@redhat.com"
##  [9] "users@redhat.com"                "users@redhat.com"
## [11] "g7MDaWX26868@hobbit.linuxworks" "users@example.com"
## [13] "tony@linuxworks.com"             "g7LKkqf15798@mail.banirh"
## [15] "users@example.com"               "admin@example.com"
## [17] "admin@example.com"               "users@example.com"
## [19] "users@example.com"               "request@example.com"
## [21] "users@example.com"               "request@redhat.com"
## [23] "request@redhat.com"              "users@redhat.com"
```

```r
faketest = extract_email_addr("i love starman by david bowie pugliese@italy.org and hello
                              walls by willie nelson blahblahblah disneyworld
                              willienelson@gmail.com pumpkinpieeeee!! #woo")
faketest
```

```
## [[1]]
## [1] "pugliese@italy.org"     "willienelson@gmail.com"
```

```r
emptytest = extract_email_addr("?? there are no emails in this test @
                               owahhhh whole wheat bread")
emptytest
```

```
## [[1]]
## character(0)
```

Exercise 4

Using your `extract_email_addr` function and the email message data:

1. How many different email addresses appear in the emails?
2. Which 5 email addresses appear the most?

YOUR ANSWER GOES HERE:

```r
alladdresses = extract_email_addr(emails)

unlistedaddresses = unlist(alladdresses)

numunlisted = length(unlistedaddresses)
numunique = unique(unlistedaddresses)

#how many all emails
numunlisted
```

```
## [1] 39446
```

```r
#how many different emails
length(numunique)
```

```
## [1] 2469
```

```r
head(sort(table(unlistedaddresses), decreasing = TRUE))
```

```
## unlistedaddresses
##         admin@xent.com       fork@example.com       request@xent.com
##                   2948                   2774                   2107
## yyyy@localhost.example          jm@jmason.org   rssfeeds@example.com
##                   1699                   1471                   1295
```

There are 2469 different email addresses in the emails.

The 5 email addresses that appear the most are:

1. admin@xent.com
2. fork@example.com
3. request@xent.com
4. yyyy@localhost.example
5. jm@jmason.org

Exercise 5

The part of an email address after the `@` is called the *domain*. The domain refers to a website, so it usually contains at least one dot. For example, Cal email addresses use the domain `berkeley.edu`, which is also the Cal website address.

Which 10 domains are the most common among the email addresses you extracted from the email data?

How many domains in the email addresses end in `.edu`?

YOUR ANSWER GOES HERE:

```r
domains = str_extract_all(unlistedaddresses, "@+[[:alpha:]]+\\.[[:alpha:]]+")
head(sort(table(unlist(domains)), decreasing = TRUE))
```

```
##
##          @example.com             @xent.com        @freshrpms.net
##                  6887                  6056                  3643
##           @jmason.org @example.sourceforge   @localhost.example
##                  2854                  2068                  1838
```

```r
domains2 = domains[which(str_detect(domains, (".edu")) == TRUE)]
edudomains = unique(unlist(domains2))
edudomains
```

```
## [1] "@ucla.edu"      "@vt.edu"        "@mit.edu"       "@unh.edu"
## [5] "@lehigh.edu"    "@dickinson.edu" "@yale.edu"      "@jhu.edu"
```

```r
length(edudomains)
```

```
## [1] 8
```

The top 10 most commons domains are:

1. @example.com
2. @xent.com
3. @freshrpms.net
4. @jmason.org
5. @example.sourceforge
6. @localhost.example
7. @lists.sourceforge
8. @redhat.com
9. @linux.ie
10. @yahoogroups.com

There are 8 email addresses that end in `.edu`.