Introduction to Working with Text Files

Shonda Kuiper

May 21, 2017

Packages Installed: stringi, knitr, ggplot2. Be noticed that we are not using the stringr package we have for the previous labs. Stringi is implemented in C and using ICU library. Stringr, which we have been using for previous labs, is based on stringi package. Stringi is faster and has more options.

Useful packages not required for this RMD:

wordcloud

readtext

[quanteda](https://cran.r-project.org/web/packages/quanteda/vignettes/quickstart.html)

#library(stringi) # finds summary information for text files

#require(tm) # Text mining

#require(RWeka) # tokenizer - create unigrams, bigrams, trigrams Depends on Java

#library(NLP); library(openNLP) #automatically installed with tm?

#library(rJava)

#library(RWekajars)

#library(SnowballC) # Stemming

#library(RColorBrewer) # Color palettes

#library(qdap)

??? What do each of these packages do? Why are they helpful? Include a comment and a link for each.

Please see Summery of useful packages for NLP

#### Reading Text Data

The goal of this activity is to start working with text files. The dataset from [SwiftKey](https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip) contains twitter, news and blog data in four languages. For this activity we'll focus on the English documents. Since these are large files, we suggest downloading onto your local computer and then loading into R.

???Better description of the data would be helpful

The readLines() function from base R can be used to read in the data. Make sure you use the proper path corresponding to your local computer {the getwd() function can be useful here}.

In order to be consistent within the labs, you need to set up a work directory for this lab.

#read in blogs, twitter and news

getwd()

setwd("C:/Users/Haoyang Liu/Desktop/Summer")

Blogs1 <- readLines("en\_US.blogs.txt", encoding = "UTF-8", skipNul = TRUE, warn = FALSE)  
Twitter1 <- readLines("en\_US.twitter.txt",encoding = "UTF-8", skipNul = TRUE, warn = FALSE)  
News1 <- readLines("en\_US.news.txt", encoding = "UTF-8", skipNul = TRUE, warn = FALSE)

It is always useful to complete a quick check to ensure each file was properly read.

#list the number of lines of each document  
length(Blogs1)  
length(Twitter1)  
length(News1)

??? Would str, dim, head, or View be better here? Why?

Dim would not work, since this is a single one character instead of table. View will try to display all the character, which is time consuming. Head can only show a few lines of data. I would recommend using both head and length.

Alternatively the readtext package works particularly well with the quanteda package. Quanteda is a fast and efficient text analysis package. Currently we need to install this readtext package directly from a repository on the github, Eventually we expect the readtext package to again be available at the cran and bioconductor repositories.

#install.packages("devtools")  
require(devtools)  
devtools::install\_github("kbenoit/readtext")   
require(readtext)  
require(quanteda)

??? Where (if at all) are these packages used in this lab?

Not for now.

#### File Size (in scripts but not in RMD files)

Before reading the data into R, you can determine basic information about the file.

file.info("en\_US.blogs.txt") ??? this only works if in the right wd  
# The size (megabites) for a particular file

The output of the file.info is in the unit of bytes. However, we want it in megabites. The conversion rate is 2^20 to 1.   
file.info("en\_US.blogs.txt")$size/1024^2 ## [1] 200.4242

#### 

#### File characteristics

Text files do not naturally fit into data frames, instead each row represents a line of text. The length of each line can vary dramatically.

??? Are all text files this format? What other formats exist?

# To view the first 3 lines  
head(Twitter1, 3)

## [1] "How are you? Btw thanks for the RT. You gonna be in DC anytime soon? Love to see you. Been way, way too long."   
## [2] "When you meet someone special... you'll know. Your heart will beat more rapidly and you'll smile for no reason."  
## [3] "they've decided its more fun if I don't."

??? Is each line an individual tweet? How do we know?

# The number of lines  
length(Twitter1) # 2360148

## [1] 2360148

# The length of the longest line  
max(nchar(Twitter1)) # 140

## [1] 140

# 5 number summary of line characters  
summary(nchar(Twitter1))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.00 37.00 64.00 68.68 100.00 140.00

The stringi package provides several convenient string/text manipulation functions. This includes summary statistics, searching for patterns, replace, split, etc... The knitr package provides a kable function that allows us to quickly create a summary table

# using the stringi package to calculate summary data (number of lines and chars)  
Bl1 <- stri\_stats\_general(Blogs1)   
Tw1 <- stri\_stats\_general(Twitter1)  
Ne1 <- stri\_stats\_general(News1)  
data1 <- data.frame(cbind(Bl1, Tw1, Ne1))  
  
#using the knitr package to make a summary table  
kable(data1, caption = "Summary Table of the Blogs, Twitter, and News documents")

If the previous one does not work, please try the following code

kable(data1, format=”pandoc”,caption = "Summary Table of the Blogs, Twitter, and News documents")

Summary Table of the Blogs, Twitter, and News documents

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bl1 | Tw1 | Ne1 |
| Lines | 899288 | 2360148 | 77259 |
| LinesNEmpty | 899288 | 2360148 | 77259 |
| Chars | 206824382 | 162096241 | 15639408 |
| CharsNWhite | 170389539 | 134082806 | 13072698 |

# Alternative Summary Table  
Lines1 <- c(length(Blogs1), length(Twitter1), length(News1))  
Words1 <- c(sum(stri\_count\_words(Blogs1)),sum(stri\_count\_words(Twitter1)),sum(stri\_count\_words(News1)))  
Chars1 <- c(sum(nchar(Blogs1)),sum(nchar(Twitter1)),sum(nchar(News1)))  
MaxWords1 <- c(max(stri\_count\_words(Blogs1)),max(stri\_count\_words(Twitter1)),max(stri\_count\_words(News1)))  
datasum2 <- data.frame(Lines1, Words1, Chars1, MaxWords1)  
names(datasum2) <- c("# Lines", "# of Words", "# of Characters","Max Words/Line")  
row.names(datasum2) <- c("Blogs", "Twitter", "News")  
datasum2

## # Lines # of Words # of Characters Max Words/Line  
## Blogs 899288 37546246 206824505 6726  
## Twitter 2360148 30093410 162096241 47  
## News 77259 2674536 15639408 1123

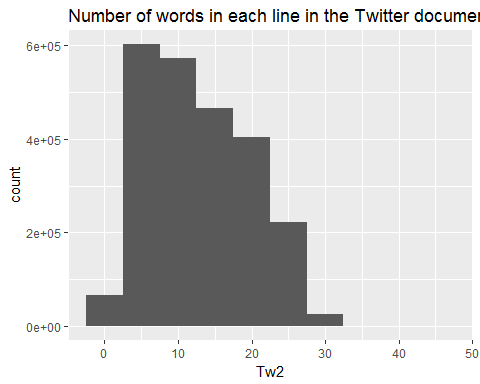
#This stringi function counts the number of words on each line in the Twitter document.   
Tw2 = stri\_count\_words(Twitter1)   
summary(Tw2)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 7.00 12.00 12.75 18.00 47.00

qplot(Tw2, binwidth = 5, main = "Number of words in each line in the Twitter document")

??? Are we more interested in counting characters instead of words in twitter?

Twitter is character-limited. We should discuss this more.



#### To read a particular line that contains a word or phrase of interest (e.g. biostats)

In Base R the grep() returns an indices vector (gives the exact line where the word or phrase is found) and grepl() returns a logical vector (a vector of TRUE or FALSE for each line of text).

# Search for a particular word or phrase in a document  
love <- sum(grepl("love", Twitter1))  
love

## [1] 90956

???This grepl was a little confusing. Can we write a little more detail?

A: grepl() returns a vector of true or false for each line of text. See example code below:

> txt <- "I love you."

> grepl("love", txt)

[1] TRUE

> txt2 <- c("I love you", "he likes you")

> grepl("love", txt2)

[1] TRUE FALSE

# R is case sensitive, to include capitol letters we list both options [], see the "Regular Expressions" document for more options.  
love <- sum(grepl("[Ll][Oo][Vv][Ee]", Twitter1))  
love

## [1] 117760

# To find the line of text that contains the word biostats  
biostats <- grep("biostats", Twitter1)  
biostats ### [1] 556,872

## [1] 556872

??? Add code for a dispersion plot HERE?

# To print the line that contains the word biostats  
Twitter1[biostats]

## [1] "i know how you feel.. i have biostats on tuesday and i have yet to study =/"

phrase1 <- grep("A computer once beat me at chess, but it was no match for me at", Twitter1)  
phrase1

## [1] 385546 519059 702303 823025 835824 837384 858020 1111563  
## [9] 1208382 1341505 1369518 1436023 1469662 1626795 1935597 1946944  
## [17] 1959503 2086880 2283423

#Print the first four lines that contain phrase1  
head(Twitter1[phrase1],4)

## [1] "A computer once beat me at chess, but it was no match for me at kick boxing. -Emo Philips"  
## [2] "A computer once beat me at chess, but it was no match for me at kickboxing"   
## [3] "A computer once beat me at chess, but it was no match for me at kick boxing."   
## [4] "A computer once beat me at chess, but it was no match for me at kick boxing."

#### ??? How do you read the shortest line? How do you sort by number of characters?

Blogs1[which.min(nchar(Blogs1))]

Twitter1[which.min(nchar(Twitter1))]

sort(nchar(Blogs1))

???We need to add some “on your own questions throughout the lab?