HW7_IS457_127

Jinran Yang

11/5/2018

HW 7 - Due Monday Nov 5, 2018 in moodle and hardcopy in class.

```
## Do not remove any of the comments. These are marked by #
## (1) Upload R file to Moodle with filename: HW7_IS457_YOURCLASSID.R
## (2) Do not remove any of the comments. These are marked by #

## For this assignment we will work with regular expressions.
## First, we will have some warmup questions, then proceed
## to learn some interesting things about Aesop's fables.
```

Part 1. Regular Expressions Warmup (12 pt)

Basic Regex

Q1. Find words in test vector test_1 which start with lowercase 'e' using grep. What does grep return? (2 pt)

```
test_1 = c("wireless", "car", "energy", "2020", "elation", "alabaster", "Endoscope")

## Your code here
grep("^[e]",test_1)
```

```
## [1] 3 5
```

```
#It returns the indices of the elements of test_1 which start with lowercase 'e'.

grep("^[e]",test_1,value = TRUE)
```

```
## [1] "energy"  "elation"
```

```
#It returns the elements of test_1 which start with lowercase 'e'.
```

Q2. Find characters which can be a password with ONLY letters and numbers. (1 pt)

```
test_2 = c("bb1l9093jak", "jackBlack3", "the password", "!h8p4$$w0rds",
           "wiblewoble", "ASimpleP4ss", "d0nt_use_this")

test_2[!grepl("[[:punct:]|[:blank:]]",test_2)]
```

```
## [1] "bb1l9093jak" "jackBlack3"  "wiblewoble"  "ASimpleP4ss"
```

Q3. Find Email addresses of the form letters@letters.xxx (1 pt)

Here "xxx" means any alpha numeric characters with length of 3. Letters can be any alpha numeric characters of any length. Letters before "@" can also be along with the underscore.

```
test_3 = c("wolf@gmail.com", "little_red_riding_hood@comcast.net", "spooky woods5@swamp.us",
           "grandma@is.eaten", "the_ax@sbcglobal.net")

## Your code here
test_3[grep("^[a-zA-Z_@a-zA-Z]*.[a-zA-Z0-9]{3}$",test_3)]
```

```
## [1] "wolf@gmail.com"
## [2] "little_red_riding_hood@comcast.net"
## [3] "the_ax@sbcglobal.net"
```

```
##Capture Groups
## This is a method to grab specific text within a given match.
## This is a very useful technique to get specific bits of text quickly.
## We will use a series of steps to extract the domain
##names from properly formatted email addresses.
```

Q4. Use regexec() to execute a regular expression to find properly formatted email addresses in test_3. Save it as test_3_reg_exec.

```
## This time, we will allow domain names of the form letters.letters.
##i.e. addresses like 'test.us' are now allowed.(1 pt)

## Your code here
test_3_reg_exec<-regexec("^[a-zA-Z_@a-zA-Z.a-zA-Z]*$",test_3)
```

Q5. What type of object is test_3_reg_exec? What type of information does it contain? (2 pt)

```
## Your code here
typeof(test_3_reg_exec)
```

```
## [1] "list"
```

```
test_3_reg_exec
```

```
## [[1]]
## [1] 1
## attr(,"match.length")
## [1] 14
## attr(,"useBytes")
## [1] TRUE
##
## [[2]]
## [1] 1
## attr(,"match.length")
## [1] 34
## attr(,"useBytes")
## [1] TRUE
##
## [[3]]
## [1] -1
## attr(,"match.length")
## [1] -1
## attr(,"useBytes")
## [1] TRUE
```

```
##
## [[4]]
## [1] 1
## attr(,"match.length")
## [1] 16
## attr(,"useBytes")
## [1] TRUE
##
## [[5]]
## [1] 1
## attr(,"match.length")
## [1] 20
## attr(,"useBytes")
## [1] TRUE
```

Your explanation here

The type of `test_3_reg_exec` is list.
The first index tells where the overall match begins; `match.length` tells the length of the matched information;The result of `attr(,"useBytes")` is logical and it tells whether the match positions and lengths are in bytes.

Q6. Use `regmatches()` to get a list of the text matches from test_3_reg_exec. Call this 'reg_match_list'.

What is the class of reg_match_list? and what is the format? (4 pt)

```
## Your code here
reg_match_list<-regmatches(test_3,test_3_reg_exec)
class(reg_match_list)
```

```
## [1] "list"
```

```
reg_match_list
```

```
## [[1]]
## [1] "wolf@gmail.com"
##
## [[2]]
## [1] "little_red_riding_hood@comcast.net"
##
## [[3]]
## character(0)
##
## [[4]]
## [1] "grandma@is.eaten"
##
## [[5]]
## [1] "the_ax@sbcglobal.net"
```

Your explanation here

The class of `reg_match_list` is list.
It returns a list contains matched data obtained by `regexec()`.

Q7. Use reg_match_list() to get a vector of matched domain names in Q6. Name this vector 'domain names'. (3 pt)

```
## Your code here
#Filter(function(x)!identical(character(0),x), reg_match_list)
domain_names<-sub(".*@","",Filter(function(x)!identical(character(0),x), reg_match_list))
domain_names
```

```
## [1] "gmail.com"     "comcast.net"   "is.eaten"      "sbcglobal.net"
```

Part 2. Aesop's Fables

We will now look at a text file of aesop's fables. We will first need to process the data to get it into a form we can use. We can then look at interesting properties like the number of words in each fable.

Q8. Use readLines() to load the aesop fable data from the aesop-fables.txt file you can find in moodle. Name it aesop_data. MAKE SURE to use the encoding 'UTF-8'. (1 pt)

```
## Your code here
aesop_data<- readLines(con <- file("./aesop-fables.txt", encoding = "UTF-8"))
close(con)
head(aesop_data)
```

```
## [1] "<U+FEFF>The Project Gutenberg EBook of Aesop's Fables, by Aesop"
## [2] ""
## [3] "This eBook is for the use of anyone anywhere at no cost and with"
## [4] "almost no restrictions whatsoever.  You may copy it, give it away or"
## [5] "re-use it under the terms of the Project Gutenberg License included"
## [6] "with this eBook or online at www.gutenberg.net"
```

Q9. What is the format of aesop_data? How is the book formatted? How might we use this formatting to our advantage? (3 pt)

Your explanation here

It is a character vector of length the number of lines aesop_data have, and each element of the character vector is one line of aesop_data. This formate is convenient for us to use regular expressions to do some text analytics.

Q10. Let's take a look of fables using the table of contents.

First, find the start point and end point of the table of content using grep() and specific header names in the file. Then subset only those lines which are from the table of contents. Save the fable titles in a character vector. Finally, count the number of non-empty lines in your subset. Print out the number.(5 pt)

```
## Your code here
start<-grep("CONTENTS",aesop_data)
end<-grep("LIST OF ILLUSTRATIONS",aesop_data)
fable_titles<-aesop_data[(start+2):(end-1)]#only fables titles without the "CONTENTS"
length(Filter(function(x)!identical("",x), fable_titles))
```

```
## [1] 284
```

Q11. Separate out all the fables in the file.

The process is similar to Q10, find the start point and end point. (3 pt) Call this fable_data. Here do not remove the titles or empty lines. NOTE: Notice that, in this text file, "AESOP'S FABLES" is sometimes shown as "??SOP'S FABLES", after you find the lines you want to extract information from, make sure you read the text carefully. (if you need to use it, just a simple copy or paste will work).

```
## Your code here
grep("??SOP'S FABLES",aesop_data)
```

```
## [1]   18   30  911 5950
```

```
start_fable<-911
end_fable<-5950
fable_data<-aesop_data[start_fable:end_fable]
```

Q12. How do you know when a new fable is starting? (1 pt)

Your explanation here

When I see a line of letters (the title of fable) which is in upper case followed by a paragraph of letters in lower case.

Q13.We will now transform this data to be a bit easier to work with.

Fables always consist of a body which contains consequtive non-empty lines which are the text of the fable. This is sometimes followed by a 'lesson' (summary) statement whose lines are consecutive but indented by four spaces. We will create a list object which contains information about each fable. Get the start positions of each fable in fable_data (how you answer Q12?). (3 pt) Hint: Look at the title vector you created in Q10, what does it include (other than letters?)

```
## Your code here
title<-Filter(function(x)!identical("",x), fable_titles)
start_position<-which(fable_data %in% title)
start_position
```

```
##   [1]    6   19   36   58   75   91  109  128  139  156  169  187  200  214
##  [15]  231  257  268  301  323  344  360  373  395  412  436  457  473  488
##  [29]  515  538  552  571  598  615  628  645  661  682  717  736  754  769
##  [43]  784  798  824  847  866  892  906  923  937  953  969  994 1012 1027
##  [57] 1045 1060 1079 1092 1116 1130 1144 1162 1177 1192 1209 1228 1240 1262
##  [71] 1280 1296 1308 1319 1333 1350 1372 1387 1403 1415 1433 1450 1463 1480
##  [85] 1502 1514 1530 1543 1560 1574 1587 1604 1620 1641 1654 1669 1692 1714
##  [99] 1729 1744 1763 1778 1796 1808 1824 1838 1858 1876 1892 1908 1923 1937
## [113] 1954 1966 1981 1996 2014 2034 2054 2073 2087 2103 2120 2134 2148 2163
## [127] 2179 2211 2234 2250 2266 2281 2295 2309 2324 2351 2365 2380 2396 2415
## [141] 2429 2457 2476 2491 2506 2522 2538 2555 2571 2589 2603 2615 2632 2650
## [155] 2668 2690 2708 2723 2740 2762 2782 2799 2815 2835 2850 2868 2887 2903
## [169] 2919 2934 2957 2975 3014 3032 3044 3064 3078 3093 3113 3133 3148 3164
## [183] 3179 3194 3214 3226 3242 3258 3277 3296 3317 3336 3356 3370 3386 3401
## [197] 3415 3436 3468 3485 3506 3527 3545 3562 3578 3595 3619 3638 3659 3681
## [211] 3697 3717 3738 3762 3777 3792 3805 3822 3840 3856 3878 3899 3913 3940
## [225] 3956 3972 3987 4001 4017 4040 4053 4064 4079 4093 4115 4128 4146 4160
## [239] 4175 4192 4208 4228 4247 4261 4282 4295 4314 4334 4354 4369 4396 4414
## [253] 4432 4449 4475 4499 4516 4527 4545 4565 4581 4597 4618 4637 4654 4668
```

```
## [267] 4680 4724 4745 4758 4771 4785 4804 4821 4837 4851 4869 4881 4904 4917
## [281] 4930 4947 4973 4988
```

Q14. Transform the fables into an easy-to-reference format (data structure).

First create a new list object named 'fables'. Each element of the list is a sublist that contains two elements ('text' and 'lesson'). For each fable, merge together the separate lines of text into a single character element. That is, one charactor vector (contains all sentences) for that fable. This will be the 'text' element in the sublist for that fable.
If the fable has a lesson, extract the statement into a character vector (also remove indentation). This will be the 'lesson' element in the sublist for that fable. (10 pt)

```
## Your code here
list1<-list(rep(0,length(title)))

       for (i in 1:(length(start_position)-1)){
   list1[[i]]<-fable_data[(start_position[i]+3):(start_position[i+1]-5)]
 }

#lesson_index<-grep("    ",list1)

fables<-list(rep(0,length(title)))

for (i in 1:length(list1)){
 if(i %in%  grep("    ",list1)) {
   fables[[i]]<-list(text=list1[[i]][1:(grep("^[ ]",list1[[i]])-2)],
                     lesson=list1[[i]][grep("^[ ]",list1[[i]]):length(list1[[i]])])
 }else{
     fables[[i]]<-list(text=list1[[i]]
                       ,lesson=list()
                       )
  }
}

for (i in 1:length(fables)){
  fables[[i]]$lesson<-sub("^[ ]{4}","",fables[[i]]$lesson)
  fables[[i]]$text<-paste(fables[[i]]$text, collapse = '')
fables[[i]]$lesson<-paste(fables[[i]]$lesson, collapse = '')
}
head(fables,n=3)
```

```
## [[1]]
## [[1]]$text
## [1] "A hungry Fox saw some fine bunches of Grapes hanging from a vine thatwas trained along a high t
##
## [[1]]$lesson
## [1] ""
##
##
## [[2]]
## [[2]]$text
## [1] "A Man and his Wife had the good fortune to possess a Goose which laida Golden Egg every day. Lu
##
## [[2]]$lesson
## [1] "Much wants more and loses all."
```

```
##
##
## [[3]]
## [[3]]$text
## [1] "There was once a house that was overrun with Mice. A Cat heard ofthis, and said to herself, \"T
##
## [[3]]$lesson
## [1] "If you are wise you won't be deceived by the innocent airs ofthose whom you have once found to
```

Q15. How many fables have lessons? (2 pt)

```
## Your code here
length(grep("    ",list1))
```

```
## [1] 89
```

Q16. Add a character count element named 'chars' and a word count element named 'words' to each fable's list. (3 pt)

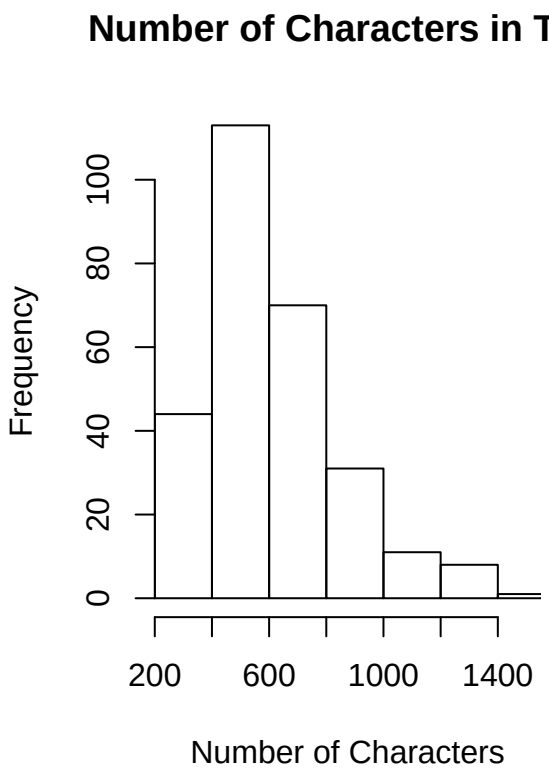
```
## Use the following function to count words:
word_count = function(x) {
  return(lengths(gregexpr("\\W+", x)) + 1)  # words separated by space(s)
}
## Your code here
for (i in 1:length(fables)){
  fables[[i]]$chars<-nchar(fables[[i]])
   fables[[i]]$words<-word_count(fables[[i]])-2
   fables[[i]]$words<-fables[[i]]$words[-3]
}
```

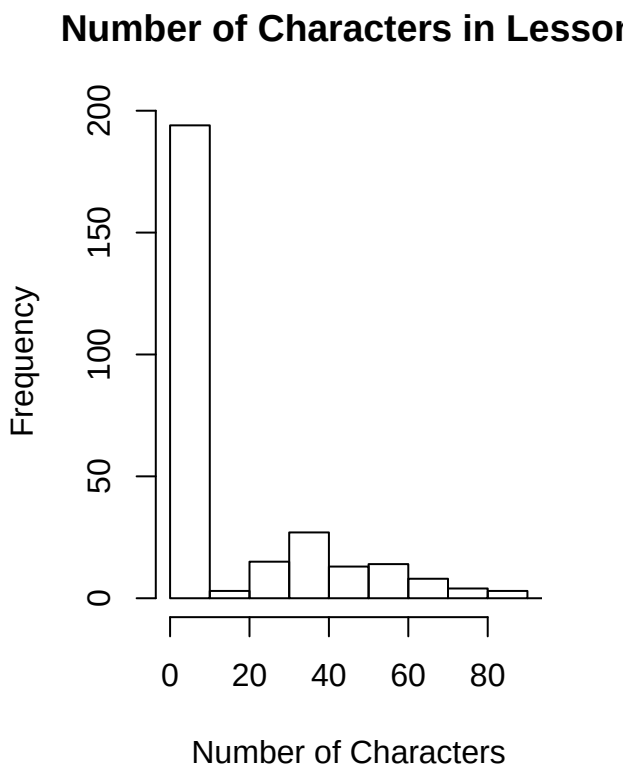
Q17. Create separate histograms of the number of characters and words in the fables. (10 pt)

Recall the graphics techniques you learned before.

```
## Your code here
######characters#######
num_chars_text<-vector(length = length((fables)))

for (i in 1: length(fables)){
 num_chars_text[i]<-fables[[i]]$chars[1]
}

num_chars_lesson<-vector(length = length((fables)))

for (i in 1: length(fables)){
 num_chars_lesson[i]<-fables[[i]]$chars[2]
}

#length(num_chars_lesson)
#quantile(num_chars_text,probs = c(0.025,0.975))
#quantile(num_chars_lesson,probs = c(0.025,0.975))

par(mfrow=c(1, 2))
```

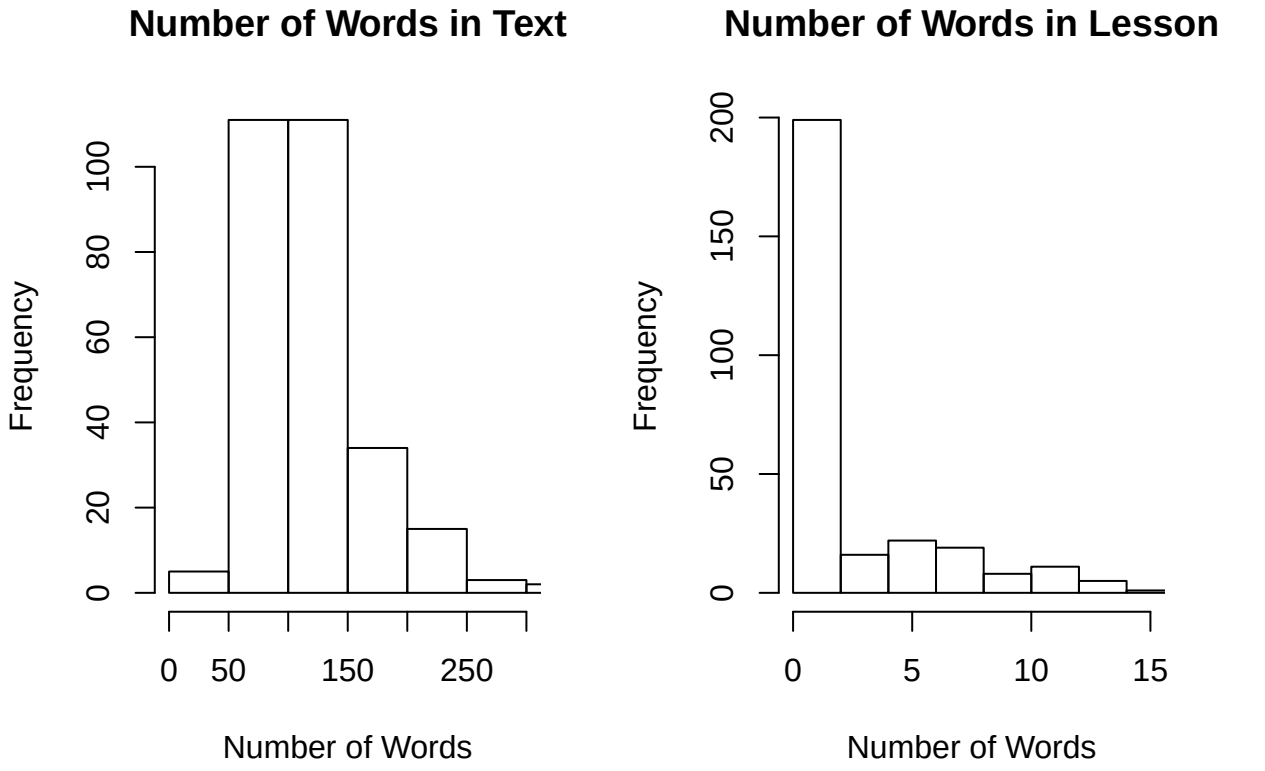
```r
hist(num_chars_text,
    xlim = c(250,1500.00),
     main = "Number of Characters in Text",
     xlab = "Number of Characters"
     )
hist(num_chars_lesson, xlim = c(0,90),
     main = "Number of Characters in Lesson",
     xlab = "Number of Characters")
```






```r
######words#######

num_words_text<-vector(length = length((fables)))

for (i in 1: length(fables)){
 num_words_text[i]<-fables[[i]]$words[1]
}

num_words_lesson<-vector(length = length((fables)))

for (i in 1: length(fables)){
 num_words_lesson[i]<-fables[[i]]$words[2]
}

#quantile(num_words_text,probs = c(0.025,0.975))
#quantile(num_words_lesson,probs = c(0.025,0.975))

par(mfrow=c(1, 2))
hist(num_words_text,
   xlim = c(0,300),
```

```
     main = "Number of Words in Text",
     xlab = "Number of Words"
     )
hist(num_words_lesson,
     xlim = c(0,15),
     main = "Number of Words in Lesson",
     xlab = "Number of Words")
```



Q18. Lets compare the fables with lessons to those without.

Extract the text of the fables (from your fables list) into two vectors. One for fables with lessons and one for those without. (4 pt)

```
## Your code here

index<-vector(length = length(fables))

for (i in 1: length(fables)){
 index[i]<-fables[[i]][3]$chars[2]==0
}

index1<-which(index == T)#position of fable without lesson
index2<-setdiff(1:length(fables),index1)#position of fable with lesson

#with_lesson<-list()
#for(j in 1:length(index2)){
#  with_lesson[[j]]<-0
#}
#for(j in 1:length(index2)){
```

```
#    with_lesson[[j]]<-list(text=fables[[index2[j]]]$text,
#                           lesson=fables[[index2[j]]]$lesson)
#}


with_lesson<-vector(length = length(index2))

for(j in 1:length(index2)){
     with_lesson[j]<-fables[[index2[j]]]$text
  }

head(with_lesson,n=2)
```

```
## [1] "A Man and his Wife had the good fortune to possess a Goose which laida Golden Egg every day. Lu
## [2] "There was once a house that was overrun with Mice. A Cat heard ofthis, and said to herself, \"T
```

```
#without_lesson<-list()
#for(j in 1:length(index1)){
#  without_lesson[[j]]<-0
#}
#  for(j in 1:length(index1)){
#    without_lesson[[j]]<-fables[[index1[j]]]$text
#  }

without_lesson<-vector(length = length(index1))

for(j in 1:length(index1)){
     without_lesson[j]<-fables[[index1[j]]]$text
  }

head(without_lesson,n=2)
```

```
## [1] "A hungry Fox saw some fine bunches of Grapes hanging from a vine thatwas trained along a high t
## [2] "There was once a Charcoal-burner who lived and worked by himself.A Fuller, however, happened to
```

Q19. Remove all non alphabetic characters (except spaces) and change all characters to lowercase. (3 pt)

```
## Your code here
########fable with lesson
with_low<-vector(length =length(with_lesson) )

for(j in 1:length(with_lesson)){
  with_low[j]<-tolower(gsub("[^[:alnum:][:blank:]]", "",with_lesson[j]))
}
head(with_low,n=2)
```

```
## [1] "a man and his wife had the good fortune to possess a goose which laida golden egg every day luc
## [2] "there was once a house that was overrun with mice a cat heard ofthis and said to herself thats
```

```
########fable without lesson

without_low<-vector(length =length(without_lesson) )

for(j in 1:length(without_lesson)){
```

```
  without_low[j]<-tolower(gsub("[^[:alnum:][:blank:]]", "",without_lesson[j]))
}
head(without_low,n=2)
```

```
## [1] "a hungry fox saw some fine bunches of grapes hanging from a vine thatwas trained along a high t
## [2] "there was once a charcoalburner who lived and worked by himselfa fuller however happened to com
```

Q20. Split the fables from Q19 by blanks and drop empty words. Save all the split words into a single list for each type of fable. Name them token_with_lessons and token_without_lessons. Print out their lengths. (5 pt)

```
## Your code here
#try<-unlist(without_low,use.names = F)
#try1<-strsplit(try," ")
#try2<-unlist(try1)

token_without_lessons<-unlist(strsplit(unlist(without_low,use.names = F)," "))
length(token_without_lessons)
```

```
## [1] 23116
```

```
token_with_lessons<-unlist(strsplit(unlist(with_low,use.names = F)," "))
length(token_with_lessons)
```

```
## [1] 10153
```

Q21. Calculate the token frequency for each type of fable. (2 pt)

```
## Your code here
freq_with<-sort(table(token_with_lessons),decreasing = TRUE)
head(freq_with,n=10)
```

```
## token_with_lessons
## the and   a  to  of  he his was you  in
## 576 447 374 308 214 210 147 135 134 132
```

```
freq_without<-sort(table(token_without_lessons),decreasing = TRUE)
head(freq_without,n=10)
```

```
## token_without_lessons
##  the  and    a   to   he   of  his  was   in  you
## 1348  896  750  723  523  509  350  319  309  303
```

Q22. Carry out some exploratory analysis of the data and token frequencies.

For example, find the words associated fables with lessons. What are distribution patterns for term frequencies?

Use wordcloud function in wordcloud package to plot your result. What are your observations? (10 pt) Hint: you'll want to include important words but not stopwords (we provided a list below) into your plot. What are important words? we have token_with(out)_lessons from Q20, think relative high frequency (use `quantile()` to help you decide). so, start by creating a table of token frequency; filter out low frequency words and stopwords.

```
mask_word = c("by", "as", "a","their", "which", "have", "with", "are", "been", "will",
              "we", "not","has", "this", "or", "from", "on", "i", "the","is","it",
```

```
              "in","my","of","to","and","be","that","for","you","but","its","was")
library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 3.4.4
```

```
## Loading required package: RColorBrewer
```

```
## Your code here:
d <- data.frame(word = names(freq_with),freq_with)#change table to data.frame
index_d<-!d$word %in% mask_word#index of not stopwords

nostop<-d[index_d,]
quantile(nostop$Freq,probs = 0.9)
```

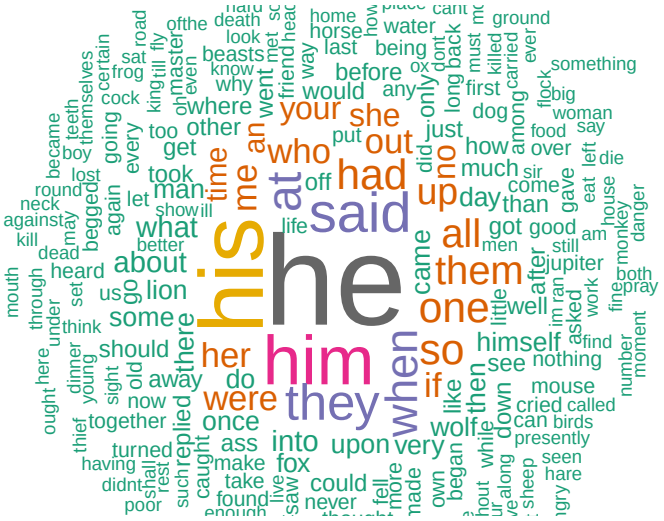
```
## 90%
##   4
```

```
wordcloud(words = nostop$word, freq = nostop$Freq, min.freq = 4,
          max.words=210, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

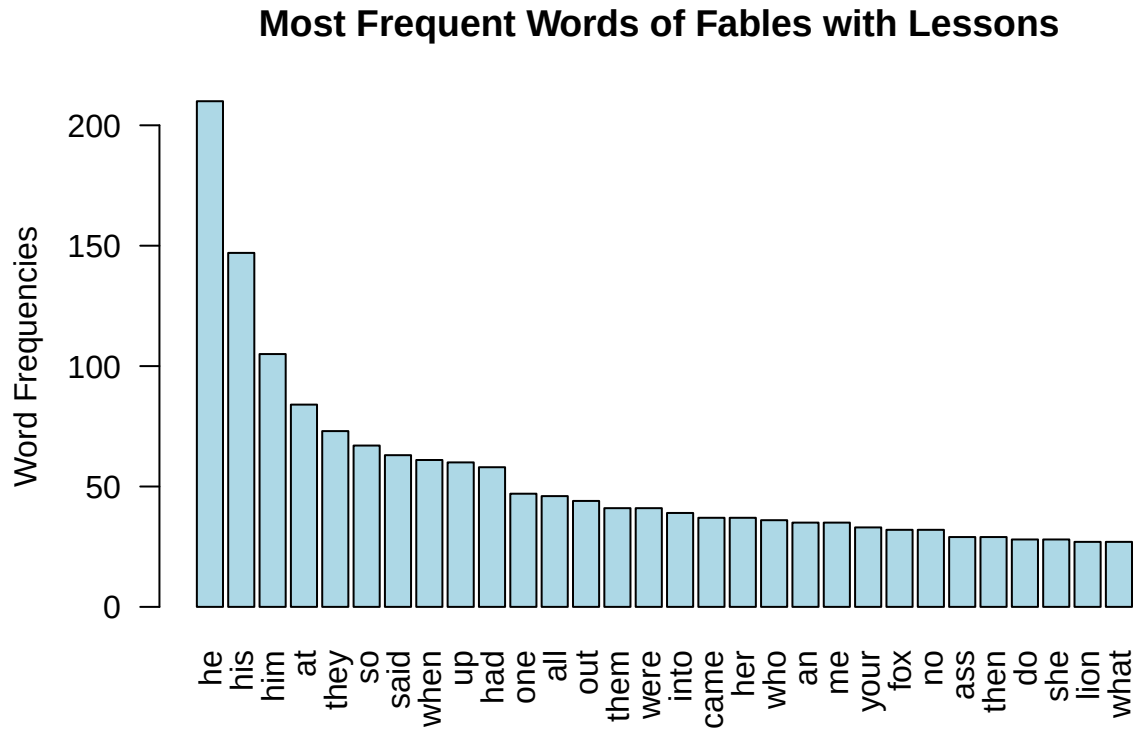

```
############fables without lesson
d_without <- data.frame(word = names(freq_without),freq_without)#change table to data.frame
index_without<-!d_without$word %in% mask_word#index of not stopwords

nostop_without<-d_without[index_without,]
quantile(nostop_without$Freq,probs = 0.95)
```

```
## 95%
##   9
```

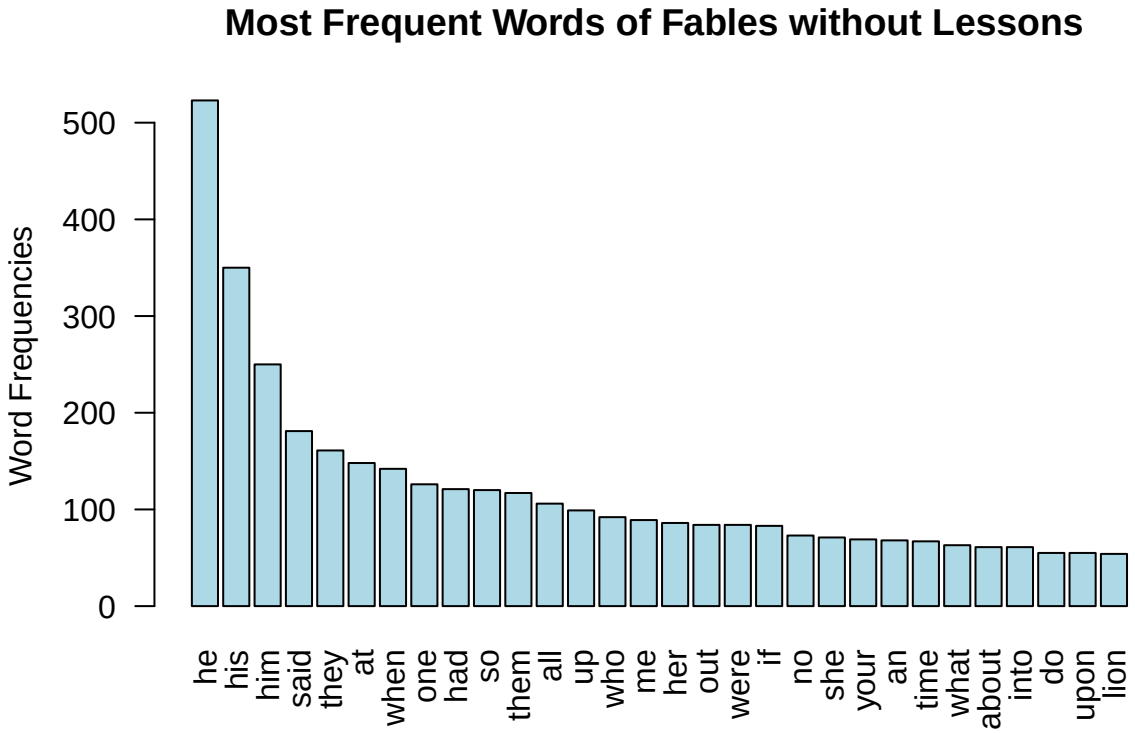
```
wordcloud(words = nostop_without$word, freq = nostop_without$Freq, min.freq = 9,
          max.words=523, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```




```
barplot(nostop[1:30,]$Freq, las = 2,names.arg = nostop[1:30,]$word,
        col ="lightblue", main ="Most Frequent Words of Fables with Lessons ",
        ylab = "Word Frequencies",
        horiz= F)
```




```
barplot(nostop_without[1:30,]$Freq, las = 2,names.arg = nostop_without[1:30,]$word,
        col ="lightblue", main ="Most Frequent Words of Fables without Lessons ",
        ylab = "Word Frequencies",
        horiz= F)
```



Your explanation here:

I plot barplots of the 30 most frequent words in texts of fables with lesson and without lesson. As we can see from both word cloud and barplots of cooresponding words frequency, there are quite a few same frequent words in fables with or without lesson. As we can see, `he`,`his`,`him` are the three most frequent words; and `she`,`at`,`they`,`when`,`all`,`one`,`them`,`up` and etc are both frequent words in two kinds of fables. This probably because fables are stories that happen on people we don't know, and it usually described in third person, so there are more subjects in the third person.