```
#############################################
### TextMining Lecture 04               #####
### Subject: Text Exploration           #####
### Developed by. KKIM                  #####
#############################################

library(dplyr)

# letters
letters
LETTERS

### toupper, tolower
c("Apple","apple","APPLE") %>%
  tolower()
c("Apple","apple","APPLE") %>%
  toupper()

c("Kim","Kimchi","Kimbob","Kim's club") %>%
  tolower()
c("Kim","Kimchi","Kimbob","Kim's club") %>%
  toupper()

### sub & gsub
sub('to','TO','K to the I to the M')
gsub('to','TO','K to the I to the M')

### nchar()
nchar('God', type='chars') #nchar('God', type='bytes')
nchar('하나님', type='chars') #nchar('하나님', type='bytes')
nchar('ㅎㅏㄴㅏㄴㅣㅁ', type='chars')
length('ㅎㅏㄴㅏㄴㅣㅁ')

nchar('God ')
nchar(' God')
nchar(' God ')
nchar('God₩t')
nchar('God₩n')
nchar('God ₩n')

### strsplit()
myconf <- 'God is good all the time.'
myconf
```

```
myconf.split <-
  strsplit(myconf, split=' ')
myconf.split
myconf.split[[1]]
myconf.split[[1]][1]

strsplit(myconf.split[[1]][1],
         split='')
strsplit(myconf.split[[1]][1],
         split=' ')

### strsplit() + unlist()
myconf <- 'God is good all the time.'
myconf.split <-
  strsplit(myconf, split=' ')
myconf.split %>% unlist()

myconf.2 <- c('God is good all the time.',
              'You are good all the time.')
myconf.2.split <-
  strsplit(myconf.2,
           split=' ')
myconf.2.split %>% unlist()

###
wiki.ds <-
"Data science is an interdisciplinary field that uses
scientific methods, processes, algorithms and systems to
extract knowledge and insights from noisy, structured and
unstructured data,and apply knowledge from data across a broad
range of application domains. Data science is related to data
mining, machine learning and big data.
Data science is a 'concept to unify statistics, data analysis,
informatics, and their related methods' in order to 'understand
and analyse actual phenomena' with data. It uses techniques and
theories drawn from many fields within the context of
mathematics, statistics, computer science, information science,
and domain knowledge. However, data science is different from
computer science and information science. Turing Award winner
Jim Gray imagined data science as a 'fourth paradigm' of
science (empirical, theoretical, computational, and now data-
driven) and asserted that 'everything about science is changing
because of the impact of information technology' and the data
```

```
deluge.
A data scientist is someone who creates programming code and
combines it with statistical knowledge to create insights from
data."

wiki.ds.para <-
  strsplit(wiki.ds, split='₩n')
wiki.ds.para
wiki.ds.para.sent <-
  strsplit(wiki.ds.para[[1]],
           split='₩₩. ')
wiki.ds.para.sent

### paste()
strsplit(myconf.split[[1]][1], split='')
paste(myconf.split[[1]][1], collapse='')

myconf.split[[1]]
paste(myconf.split[[1]], collapse='')
paste(myconf.split[[1]], collapse=' ')

### Exercise
hgu <- 'Why not change the world?'

# (1-1) Count the number of spaces in a variable called  'hgu'
and
# save it to a variable called  'nspace'
nspace <-
  strsplit(hgu, split=" ")[[1]] %>%
  length - 1

# (1-2) Using the for statement, create a list called
 'hgu.str'
# which contains each word for each list element
hgu.str <- list()
for(i in 1:(nspace+1)){
  hgu.str[[i]] <- strsplit(hgu, split=' ')[[1]][i]
}
hgu.str

# (1-3) Using  'hgu.str' , create a string vector,
#  "Why not change the world?"
paste(hgu.str, collapse=' ')
```

Exercise
love <-
"Love encompasses a range of strong and positive emotional and mental states, from the most sublime virtue or good habit, the deepest interpersonal affection, to the simplest pleasure. An example of this range of meanings is that the love of a mother differs from the love of a spouse, which differs from the love for food. Most commonly, love refers to a feeling of a strong attraction and emotional attachment.
Love is considered to be both positive and negative, with its virtue representing human kindness, compassion, and affection, as 'the unselfish loyal and benevolent concern for the good of another' and its vice representing human moral flaw, akin to vanity, selfishness, amour-propre, and egotism, as potentially leading people into a type of mania, obsessiveness or codependency. It may also describe compassionate and affectionate actions towards other humans, one's self, or animals. In its various forms, love acts as a major facilitator of interpersonal relationships and, owing to its central psychological importance, is one of the most common themes in the creative arts. Love has been postulated to be a function that keeps human beings together against menaces and to facilitate the continuation of the species.
Ancient Greek philosophers identified six forms of love: essentially, familial love (in Greek, Storge), friendly love or platonic love (Philia), romantic love (Eros), self-love (Philautia), guest love (Xenia), and divine love (Agape). Modern authors have distinguished further varieties of love: unrequited love, empty love, companionate love, consummate love, infatuated love, self-love, and courtly love. Numerous cultures have also distinguished Ren, Yuanfen, Mamihlapinatapai, Cafuné, Kama, Bhakti, Mettā, Ishq, Chesed, Amore, Charity, Saudade (and other variants or symbioses of these states), as culturally unique words, definitions, or expressions of love in regards to a specified 'moments' currently lacking in the English language.
Scientific research on emotion has increased significantly over the past two decades. The color wheel theory of love defines three primary, three secondary and nine tertiary love styles, describing them in terms of the traditional color wheel. The triangular theory of love suggests 'intimacy, passion and commitment' are core components of love. Love has additional

```
religious or spiritual meaning. This diversity of uses and
meanings combined with the complexity of the feelings involved
makes love unusually difficult to consistently define, compared
to other emotional states."

# Create a word vector called  “love.para.sent.word”
love.para <-
  strsplit(love,
           split='₩n')
love.para.sent <-
  strsplit(love.para[[1]],
           split='₩₩. ')
love.para.sent.word <- list()
for(i in 1:length(love.para.sent)){
  love.para.sent.word[[i]] <-
    strsplit(love.para.sent[[i]], split=' ')
}

# What is the fourth paragraph’ s last sentence?
paste(love.para.sent.word[[4]]
[[length(love.para.sent.word[[4]])]],collapse=' ')

# What is the second paragraph’ s second sentence’ s first
word?
love.para.sent.word[[2]][[2]][1]

########################################
### Text Indexing                    ###
########################################
### regexpr & gregexpr

regexpr('to',
        'K to the l to the M')
regexpr('to', 'K to the l to the M')[1]

regexpr('to',
        c('tomorrow',
          'K to the l to the M'))

attr(regexpr('to', 'K to the l to the M'), "match.length")
attr(regexpr('to', 'K to the l to the M'), "index.type")
attr(regexpr('to', 'K to the l to the M'), "useBytes")
```

```r
gregexpr('to', 'K to the I to the M')
gregexpr('to', 'K to the I to the M')[[1]][1]
attr(gregexpr('to', 'K to the I to the M')[[1]],
"match.length")

gregexpr('to',
        c('tomorrow',
          'K to the I to the M'))

regexpr('KIM', 'K to the I to the M')
gregexpr('KIM', 'K to the I to the M')

### regexec
regexec('to', 'K to the I to the M')
regexec('to', 'K to the I to the M')[[1]][1]
attr(regexec('to', 'K to the I to the M')[[1]], "match.length")

# ()
regexec('t(o)', 'K to the I to the M')
regexec('th(e)', 'K to the I to the M')

# regexpr comparison
regexpr('t(o)', 'K to the I to the M')
regexpr('th(e)', 'K to the I to the M')

regexec('KIM', 'K to the I to the M')

### grep
grep('to', c('me','K to the I to the M'))
grepl('to', c('me','K to the I to the M'))
# lapply(wiki.ds.para.sent, length)

grep('data', wiki.ds.para.sent)
grepl('data', wiki.ds.para.sent)

grep('data', wiki.ds.para.sent[[1]])
grepl('data', wiki.ds.para.sent[[1]])

### regmatches()
kim.regexpr <- regexpr('to', 'K to the I to the M')
kim.regexpr
regmatches('K to the I to the M', kim.regexpr)
```

```
kim.gregexpr <- gregexpr('to', 'K to the I to the M')
kim.gregexpr
regmatches('K to the I to the M', kim.gregexpr)

regmatches('K to the I to the M', kim.regexpr, invert=FALSE)
regmatches('K to the I to the M', kim.regexpr, invert=TRUE)
regmatches('K to the I to the M', kim.gregexpr, invert=FALSE)
regmatches('K to the I to the M', kim.gregexpr, invert=TRUE)

strsplit('K to the I to the M', split='to')

### Text Indexing with Patterns
ex.gregexpr <- gregexpr('to', 'I want to go to home tomorrow')
regmatches('I want to go to home tomorrow', ex.gregexpr)

ex.gregexpr.1 <- gregexpr('to ', 'I want to go to home
tomorrow')
regmatches('I want to go to home tomorrow', ex.gregexpr.1)

to.gregexpr <- gregexpr('to', 'I want to eat tomato today')
regmatches('I want to eat tomato today', to.gregexpr)
to.gregexpr.1 <- gregexpr('to ', 'I want to eat tomato today')
regmatches('I want to eat tomato today', to.gregexpr.1)
to.gregexpr.2 <- gregexpr(' to ', 'I want to eat tomato today')
regmatches('I want to eat tomato today', to.gregexpr.2)

sample <- 'Amazing Korean singer is singing on the stage'
ing.gregexpr <- gregexpr('ing',sample)
regmatches(sample, ing.gregexpr)

ing.gregexpr.1 <- gregexpr('ing ',sample)
regmatches(sample, ing.gregexpr.1)

# patterns
ing.gregexpr.2 <- gregexpr('[[:alpha:]]ing',sample)
regmatches(sample, ing.gregexpr.2)

ing.gregexpr.3 <- gregexpr('[[:alpha:]]+ing',sample)
regmatches(sample, ing.gregexpr.3)

ing.gregexpr.4 <- gregexpr('[[:alpha:]]+(ing)₩₩b',sample)
regmatches(sample, ing.gregexpr.4)
```

```
### Obama Speech
# install.packages("officer")
library("officer")

obama.speech <-
  read_docx("R file/R file_LEC04/Obama_2004_speech.docx")
obama.speech

obama.speech.sum <-
  obama.speech %>%
  docx_summary
obama.speech.sum %>% head(2)
obama.speech.sum %>%
  select(content_type) %>%
  unique
obama.speech.sum %>% nrow

obama.speech.sum$text %>% length # 45 paragraphs

### Exercise
# Find out in which paragraph the word ‘free’ is used
regexpr('free',
        obama.speech.sum$text)

# Check how the word ‘free’ is used in each case.
which(regexpr('free',
              obama.speech.sum$text) != -1)
obama.speech.sum$text[3]
obama.speech.sum$text[22]
obama.speech.sum$text[35]

# regmatches + regexpr
# Returns first matched term
obama.speech.sum$text %>%
  regmatches(regexpr('free', obama.speech.sum$text),
             invert=FALSE)
# regmatches + gregexpr
# Returns list of matched firms
# In this case, each paragraph's
# matched words are presented with list
obama.speech.sum$text %>%
  regmatches(gregexpr('free', obama.speech.sum$text),
             invert=FALSE)
```

```
# How many times did the word “America” has been used in the
speech?
regexpr('America',
        obama.speech.sum$text)
obama.speech.sum$text %>%
  regmatches(gregexpr('America',
                      obama.speech.sum$text),
             invert=FALSE)

# WRONG!!
obama.speech.sum$text %>%
  regmatches(gregexpr('America',
                      obama.speech.sum$text),
             invert=FALSE) %>% length
obama.speech.sum$text %>%
  regmatches(gregexpr('America',
                      obama.speech.sum$text),
             invert=FALSE) %>% unlist %>% length
obama.speech.sum$text %>%
  regmatches(gregexpr('America', obama.speech.sum$text),
             invert=FALSE) %>% lengths %>% sum


######################################
### stringr package                ###
######################################

library(stringr)

### str_extract() & str_extract_all()
wiki.ds.short <-
  "Data science is a 'concept to unify statistics, data
analysis, informatics, and their related methods' in order to
'understand and analyse actual phenomena' with data."
str_extract(wiki.ds.short,
            "Data")
str_extract(wiki.ds.short,
            "data")
str_extract(tolower(wiki.ds.short),
            "data")
str_extract_all(tolower(wiki.ds.short),
                "data")
```

```
regexpr("data",
        tolower(wiki.ds.short))
gregexpr("data",
         tolower(wiki.ds.short))

regmatches(wiki.ds.short,
           gregexpr("data", tolower(wiki.ds.short)))

### str_count
wiki.ds.short
str_count(wiki.ds.short, " ")
str_count(wiki.ds.short, "data")
str_count(tolower(wiki.ds.short), "data")

### str_detect
wiki.ds.short
str_detect(wiki.ds.short, " ")
str_detect(wiki.ds.short, "DATA")
str_detect(toupper(wiki.ds.short), "DATA")

### str_split()
str_split(wiki.ds.short, " ")
strsplit(wiki.ds.short, " ")
strsplit(wiki.ds.short, "'")

### str_split_fixed()
library(dplyr)
library(stringr)
wiki.ds.short.5 <-
  str_split_fixed(wiki.ds.short, " ", 5)

col.count <- 4
for(i in 1:(str_count(wiki.ds.short, " ")/col.count)){
  if (i == 1){
    temp <- str_split_fixed(wiki.ds.short, " ", col.count+1)
    wiki.ds.short.2d <- temp[1:col.count]
  } else{
    temp.0 <- str_split_fixed(temp[col.count+1], " ",
col.count+1)
    wiki.ds.short.2d <- rbind(wiki.ds.short.2d,
temp.0[1:col.count])
    temp <- str_split_fixed(temp[col.count+1], " ",
col.count+1)
  }
  print(i)
}
wiki.ds.short.2d %>% data.frame
wiki.ds.short.2d
t(wiki.ds.short.2d)

### str_replace & str_replace_all
apple <- "The CEO of Apple Incorporation loves eating an apple
and apple pie"

sub('Apple', 'apple', apple)
gsub('apple', 'APPLE', apple)

apple %>% sub('Apple', 'apple')

str_replace(apple, 'Apple', 'apple')
str_replace_all(apple, 'apple', 'APPLE')


# Useful when using pipe operator
apple %>% tolower %>%
  sub('Apple', 'apple')

apple %>% tolower %>%
  str_replace('apple','')
apple %>% tolower %>%
  str_replace_all('apple','')

apple %>% tolower %>%
  str_remove('apple')
apple %>% tolower %>%
  str_remove_all('apple')

# Useful when handling unique text and removing
# meaningless text
apple %>%
  str_replace('Apple Incorporation', 'Apple_Incorporation') %>%
  str_split(" ")
apple %>%
  str_replace_all('Apple', 'zzzAPPLEzzz') %>%
  str_split(" ")
```

```
apple %>%
  str_replace('Apple Incorporation', 'Apple_Incorporation') %>%
  str_split(" ") %>% unlist
apple %>%
  str_replace_all('Apple', 'zzzAPPLEzzz') %>%
  str_split(" ") %>% unlist

apple %>%
  str_split(" ") %>%
  str_replace_all('Apple Incorporation',
                  'Apple_Incorporation')
apple %>%
  str_split(" ") %>%
  str_replace_all('Apple Incorporation', 'AppleIncorporation')

apple %>%
  str_split(" ") %>% .[[1]] %>%
  str_replace_all('Apple', 'Apple_Incorporation')
apple %>%
  str_split(" ") %>% .[[1]] %>%
  str_replace_all('Apple Incorporation', 'AppleIncorporation')

### str_remove
apple %>%
  str_remove('apple') %>%
  str_split(" ")

apple %>%
  str_remove_all('apple') %>%
  str_split(" ")

apple %>%
  str_remove_all('apple ') %>%
  str_split(" ")

### str_glue
user.name <- "김경외"
card.name <- "Hyundai-9406"
str_glue("Hi {user.name},Your payment from {card.name} was
declined.
         To keep your subscription active, please change the
payment method.")
```

```
df.set <- data.frame(user.name,card.name)
df.set
str_glue("Hi {df.set$user.name},Your payment from
{df.set$card.name} was
         declined. To keep your subscription active, please
change the payment
         method.")
```