stringr 패키지와 문자열 관련 함수

참고 : http://www.datamarket.kr/xe/board_BoGi29/12682

```
> install.packages("stringr")
> library(stringr)
```

함수	설명	예
str_detect()	주어진 데이터들에서 특정 문자가 있는지 검사해서 TRUE/FALSE를 출력	> v<-c('apple','Apple','banana','grape') > str_detect(v, 'A') # 'A'가 있는 문자열 검사 [1] FALSE TRUE FALSE FALSE > str_detect(v, 'a') [1] TRUE FALSE TRUE TRUE
str_extract()	패턴과 일치하는 부분의 문자열을 추출하는 함수 • str_extract(문자열,추출하고 싶은 패턴) ※ grep과의 차이점 : grep 함수는 패턴과 일치하는 경우 원래의 형태로 돌려줍니다.	shopping_list <- c("apples x4", "flour", "sugar", "milk x2") str_extract(shopping_list, "₩₩d") [1] "4" NA NA "2" > str_extract(shopping_list, "[a-z]+") > [1] "apples" "flour" "sugar" "milk" ## grep > grep("₩₩d", shopping_list, value = TRUE) [1] "apples x4" "milk x2" > grep("[a-z]+", shopping_list, value = TRUE) [1] "apples x4" "flour" "sugar" "milk x2"
str_sub()	주어진 index의 문자들을 추출 • str_sub(문자열, start=시작숫자, end=끝숫자) ※ base::substr()와 같은 기능을 하는 함수	> str_sub("Statistics", 2, 5) [1] "tati"
str_replace	매치되지 않은 부분은 그대로 두고 매치된 부분만 치환하는 함수 • str_replace(문자열,매치할 부분,치환할 문자) ※base::sub(매치할 부분,치환할 문자,문자열)와 같은 기능을 합니다.	> fruits <- c("one apple", "two pears", "three bananas") > str_replace(fruits, "[aeiou]", "-") #> [1] "-ne apple" "tw- pears" "thr-e bananas"
str_count()	주어진 데이터에 해당 글자가 몇번 나오는지 알려줌	> v<-c('apple','Apple','banana','grape') > str_count(v, 'a') [1] 1 0 3 1
str_length()	문자열의 길이를 계산해주는 함수 ※ base::nchar(x)와 같은 기능을 하는 함수	> v<-c('apple','Apple','banana','grape', NA) > str_length(v) [1] 5 5 6 5 NA
str_split()	주어진 데이터셋을 지정된 기호로 분리 (list형태) ※ strsplit()와 같은 기능을 하는 함수	> str_split("Happy-Holiday", "-") [[1]] [1] "Happy" "Holiday" > str_split(c("Happy-Holiday", "Hello-World"), "-") [[1]] [1] "Happy" "Holiday"

		[[2]] [1] "Hello" "World"
str_c()	문자열을 결합 • str_c(문자열,sep=",collapse=) - sep와 collapse의 차이는 한 벡터 안에 존재 하느냐의 차이 ※ base::paste0()와 비슷	> str_c(letters, "-", 1:26) # length()값은 26 [1] "a-1" "b-2" "c-3" "d-4" "e-5" "f-6" "g-7" "h-8" "i-9" "j-10" "k-11" [12] "l-12" "m-13" "n-14" "o-15" "p-16" "q-17" "r-18" "s-19" "t-20" "u-21" "v-22" [23] "w-23" "x-24" "y-25" "z-26" > str_c(letters, collapse = ",") # length()값은 1 [1]"a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z"
str_trim()	공백문자를 제거하는 함수 str_trim(문자열, side= "")	str_trim(" 바보 ", side="left") [1] "바보 " str_trim(" 바보 ", side="right") [1] " 바보" > str_trim(" 바보 ", side="both") #> [1] "바보"

[:alnum:]

Alphanumeric characters: [:alpha:] and [:digit:].

[:alpha:]

Alphabetic characters: [:lower:] and [:upper:].

[:blank:]

Blank characters: space and tab, and possibly other locale-dependent characters such as non-breaking space.

[:cntrl:]

Control characters. In ASCII, these characters have octal codes 000 through 037, and 177 (DEL). In another character set, these are the equivalent characters, if any.

[:digit:]

Digits: 0 1 2 3 4 5 6 7 8 9.

[:graph:]

Graphical characters: [:alnum:] and [:punct:].

[:lower:]

Lower-case letters in the current locale.

[:print:]

Printable characters: [:alnum:], [:punct:] and space.

[:punct:]

Punctuation characters:

! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [₩] ^ _ ` { | } ~.

[:space:]

Space characters: tab, newline, vertical tab, form feed, carriage return, space and possibly other locale-dependent characters.

[:upper:]

Upper-case letters in the current locale.

[:xdigit:]

Hexadecimal digits:

0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f.

ADP 6회 유사 문제 만들기

ADP6회	현재 문제
- 데이터 : location.txt (UTF-8, 헤더 없음) 가평,ncn 가야,ncn 남이섬,ncn .... - 데이터 : blog.txt (TAB문자로 구분됨) DATE TITLE CONTENT 20150101 제목 봄관련 내용.... .... (가) 자료읽기 (ㄱ) location.txt를 읽어서 사용자사전에 등록하기 (ㄴ) blog.txt를 다음 형식으로 읽을 것 DATE : numeric TITLE : character CONTENT : character (나) blog.txt에서 봄여행,벚꽃축제,봄나들이 등 봄과 관련된 문서만 추출하기 - (가)에서 읽은 사용자 사전에 들어있는 지명이 들어있는 문서만 추출 (다) 위에서 추출된 문서에 대해 명사추출 및 출현 빈도 10위 추출 (라) 봄과 관련된 지명 출현 빈도 10위까지 추출하여 시각화	- 데이터 : jeju_location.txt (UTF-8, 헤더 없음) - 데이터 : trip.csv

```
1  ######################
2  #
3  # 여행
4  #
5  ######################
6  library(stringr)
7  library(KoNLP)
8  library(wordcloud)
9  options(mc.cores=1)
10 ######################
11 #  함수
12 ######################
13 get_top10 <- function(data){
14         ## 추출된 문서에 대해 명사추출 및 출현 빈도 10위 추출
15         noun = extractNoun(data)
16         noun = unlist(noun)
17         noun<-gsub("₩₩d+","",noun) #숫자 제거
18         #noun<-gsub(" ","",noun) # 공백 제거
19         noun<-gsub("₩₩.","",noun) # 마침표 제거
20         noun<-gsub("₩₩^","",noun) # 꺽쇠들어간 글자 제거
21         noun <- gsub("[a-zA-Z]", "", noun) # 알파벳제거
22         noun = Filter(function(x){nchar(x)>=2 & nchar(x) <=5}, noun)
23         noun
24         noun.s = sort(table(noun), decreasing = TRUE)
```

```
        head(noun.s, 10)
}

## 파일읽기
trip.org = read.csv("trip.csv", stringsAsFactors = FALSE)

## 1. 장소
location = read.csv("jeju_location.txt", header=FALSE, stringsAsFactors = FALSE)
useNIADic()
### 2. 장소를 사전에 등록
buildDictionary(ext_dic ="woorimalsam")
buildDictionary(ext_dic = "woorimalsam",user_dic=location, replace_usr_dic = T)
get_dictionary('user_dic') # 추가여부 확인

## 제주와 관련된 문서만 추출하기
title_idx = str_detect(trip.org$title, "제주")
content_idx = str_detect(trip.org$contents, "제주")
comment = trip.org[title_idx | content_idx, ]$contents
get_top10(comment)

## 지명 출현 빈도 10위까지 추출하여 시각화
idx = rep(0, length(trip.org$contents))
for(i in 1:length(location$V1)){
        cur_idx = str_detect(trip.org$contents, location$V1[i])
        idx = idx | cur_idx
}
comment = trip.org[idx, ]$contents
wordcount = get_top10(comment)
wordcloud(names(wordcount), freq=wordcount, scale=c(7,.5),
          rot.per=0.25, min.freq=1, random.order = F,
          random.color = T, colors = brewer.pal(5,"Set2"))
```