String

박찬영

2024-08-26

tidyverse와 stringr library를 사용합니다. 문자열을 다룰 때는 stringr이 좋다. (tidyverse에 없음)

문자열 다루기

```
#문자열은 따옴표로 묶기가 기본
str1 = "This is a string"
str2 = '인용문이 있으면 "작은따옴표를 쓴다" 크크'
#따옴표 기호를 쓰고싶으면 \를 사용해라
str="따옴표 출력 하고 싶어요 \" 이렇게~"
"이거는 백슬래시 출력이에요 \\ 이렇게"
```

```
## [1] "이거는 백슬래시 출력이에요 \\ 이렇게"
```

```
#근데 터미널에서 \도 잘보이는데요

str_view(str) #얘가 진짜 출력이에요~
```

```
## [1] | 따옴표 출력 하고 싶어요 " 이렇게~
```

```
r"(얘를 쓰면요~ \ " " ? 이딴거 다 돼요~)"
```

```
## [1] "얘를 쓰면요~ \\ \" \" ? 이딴거 다 돼요~"
```

```
str_view("크크 \n \t 이런건 다 알죠?")
```

```
## [1] | 크크
##     |  {\t} 이런건 다 알죠?
```

```
"\uc804\uc0b0\ud1b5\uacc4" #유니코드 출력법
```

```
## [1] "전산통계"
```

```
length("문자열의 길이") #는 1이다 !
```

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

```r
str_length("이게 ㄹㅇ 문자열 길이")
```

```
## [1] 12
```

```r
str_length(c("하","하하",NA))
```

```
## [1]  1  2 NA
```

```r
str_c("x", "y") #기본 결합
```

```
## [1] "xy"
```

```r
str_c("x","y",sep="..") #구분자
```

```
## [1] "x..y"
```

```r
str_c("나는",c("박찬영", "박찬빵"), "입니다.", sep=" ")
```

```
## [1] "나는 박찬영 입니다." "나는 박찬빵 입니다."
```

```r
#이러면 벡터로 결합된다 굿굿



str_c(
  "Good ", "morning", " ", "chan",
  if (FALSE) " and HAPPY BIRTHDAY",
  "."
)
```

```
## [1] "Good morning chan."
```

```r
#if문 사용으로 문자열을 선택할 수 있음


str_c(c("나는", "나는","저팔계"), collapse = " ")
```

```
## [1] "나는 나는 저팔계"
```

```r
#collapse를 쓰면 문자열 벡터를 합칠 수 있다


#문자열을 슬라이싱 해보자


str_sub("Apple", 1,3) #1에서 3까지
```

```
## [1] "App"
```

```r
str_sub(c("나는야","너는야","저팔계야"), 1,2)
```

```
## [1] "나는" "너는" "저팔"
```

```r
str_sub(c("나는야","너는야","저팔계야"), -2,-1)
```

```
## [1] "는야" "는야" "계야"
```

```r
#음수 사용하면 뒤로간다

x <- c("Apple", "Banana", "Pear")
str_sub(x, 1, 1) <- str_to_lower(str_sub(x, 1, 1))
x
```

```
## [1] "apple"  "banana" "pear"
```

```r
#부분 할당이 된다...
```

정규 표현식

정규표현식은 꽤 어렵다… 일단 매칭탐지부터 가자

```r
x=c("apple","banana","pear")

str_detect(x, "e") #각 요소별로 e가 들어가는지 확인
```

```
## [1]  TRUE FALSE  TRUE
```

```r
#논리형 벡터 반환이라 filter에 잘 쓰인다

library(babynames)
```

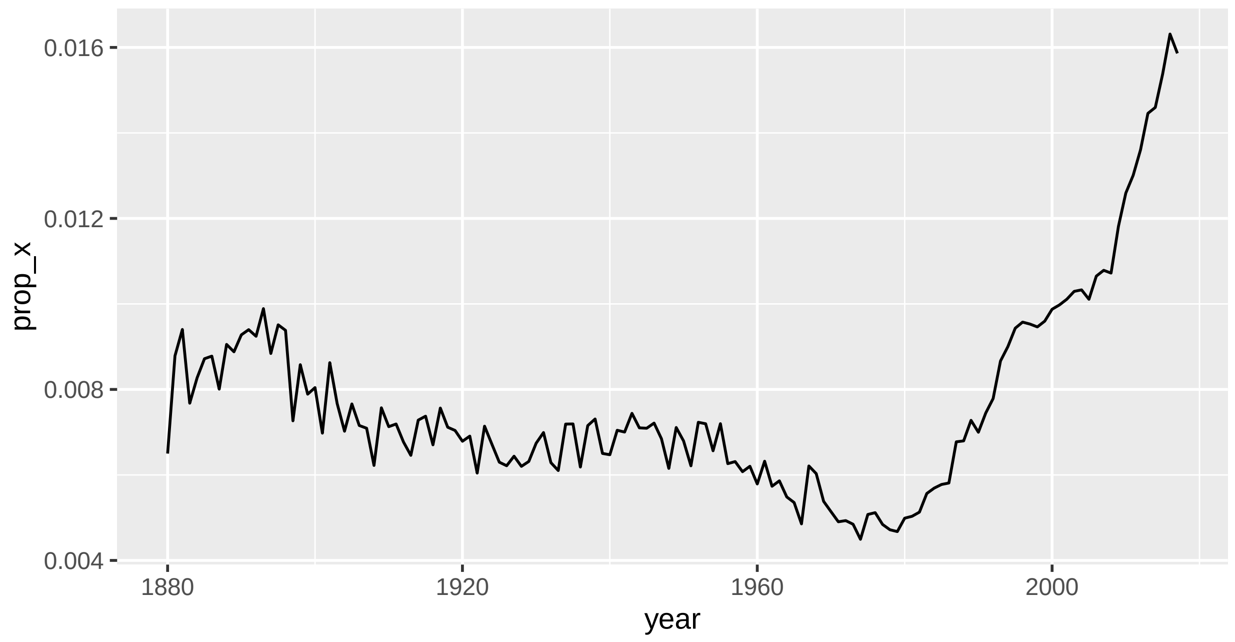
```
## Warning: package 'babynames' was built under R version 4.4.1
```

```r
babynames %>% filter(str_detect(name, "x"))
```

```
## # A tibble: 16,317 x 5
##     year sex   name          n      prop
##    <dbl> <chr> <chr>     <int>     <dbl>
##  1  1880 F     Roxie        62 0.000635
##  2  1880 F     Dixie        15 0.000154
##  3  1880 F     Roxanna       9 0.0000922
##  4  1880 F     Texas         5 0.0000512
##  5  1880 M     Alexander   211 0.00178
##  6  1880 M     Alex        147 0.00124
##  7  1880 M     Felix        92 0.000777
##  8  1880 M     Max          52 0.000439
## 9   1880 M     Axel          16 0.000135
## 10  1880 M     Rex           13 0.000110
## # i 16,307 more rows
```

```
#논리 벡터이기 때문에 sum은 일치수 mean은 일치율을 알려준다

babynames %>%
    group_by(year) %>%
    summarise(prop_x = mean(str_detect(name, "x"))) %>%
    ggplot(aes(year, prop_x)) +
    geom_line() +
    theme(aspect.ratio = 1/2)
```



```
#str_detect는 기본이 정확한 포함이지만 여러가지 용법이 있다

str_detect(c("a","ab","ea"), "a.")
```

```
## [1] FALSE  TRUE FALSE
```

```
#a. 은 a뒤에 뭔가가 오는 것을 의미한다 a를 포함이 아니다
```

```r
str_detect(x, ".a.") #이러면 a앞뒤에 한 글자씩 있어야 함
```

```
## [1] FALSE  TRUE  TRUE
```

```r
str_view(x, ".a.") #얘는 시각적 피드백을 준다
```

```
## [2] | <ban>ana
## [3] | p<ear>
```

```r
str_view(x, "^a") #a로 시작하는 거 찾기
```

```
## [1] | <a>pple
```

```r
str_view(x, "a$") #a로 끝나는 거 찾기
```

```
## [2] | banan<a>
```

```r
y <- c("apple pie", "apple", "apple cake", "pine apple")


str_view(y, "^apple$") #강제하기
```

```
## [2] | <apple>
```

```r
str_view(y, "apple$")
```

```
## [2] | <apple>
## [4] | pine <apple>
```

```r
str_view(c("a","ab","abb","abcc"),"abc?") #하면 c는 선택사항이 된다 ab는 필수
```

```
## [2] | <ab>
## [3] | <ab>b
## [4] | <abc>c
```

```r
str_view(c("a","ab","abb","abbb"),"ab+") #하면 b를 더할 수 있다 ab는 필수
```

```
## [2] | <ab>
## [3] | <abb>
## [4] | <abbb>
```

```r
str_view(c("a","ab","abbb","abcabbb"),"ab*") #하면 b는 선택사항이고 반복가능 +와 ?의 합
```

```
## [1] | <a>
## [2] | <ab>
## [3] | <abbb>
## [4] | <ab>c<abbb>
```

```r
names=c("Hadley","Mine","Garrett")


str_view(names, "[aeiou]") #[]는 안에 있는거를 다 찾음
```

```
## [1] | H<a>dl<e>y
## [2] | M<i>n<e>
## [3] | G<a>rr<e>tt
```

```r
str_view(names, "[^aeiou]") #[]안의 ^는 제외
```

```
## [1] | <H>a<d><l>e<y>
## [2] | <M>i<n>e
## [3] | <G>a<r><r>e<t><t>
```

```r
str_view(names, "[a^eiou]") #이런건 안됨ㅋㅋㅋ
```

```
## [1] | H<a>dl<e>y
## [2] | M<i>n<e>
## [3] | G<a>rr<e>tt
```

```r
#핵심은 각각 찾아준다는 것
str_view(names, "[^aeiou]+") #+랑 합치면 이렇게됨
```

```
## [1] | <H>a<dl>e<y>
## [2] | <M>i<n>e
## [3] | <G>a<rr>e<tt>
```

```r
#연결해서 찾아줌 (반복 허용이라)


str_view(x,"p{2}") #{n} 은 갯수
```

```
## [1] | a<pp>le
```

```r
str_view(x, "p{1,2}") #{n,m} 은 n이상 m이하
```

```
## [1] | a<pp>le
## [3] | <p>ear
```

```r
str_view(x, "p{1,}") #1, 하면 이상
```

```
## [1] | a<pp>le
## [3] | <p>ear
```

```r
str_view(x, "p{1,2}?") #뭔 작동이여
```

```
## [1] | a<p><p>le
## [3] | <p>ear
```

```
str_view(x,"a[pr]+?") #이런 거?도 몰루
```

```
## [1] | <ap>ple
## [3] | pe<ar>
```

```
str_view(x,"(..)\\1") #\n 은 정규표현식인데 문자열이라서 \\n으로 써야하고 앞의 그룹표현식에대한 참조임
```

```
## [2] | b<anan>a
```

```
str_view(x,"(a.)\\1")
```

```
## [2] | b<anan>a
```

```
head(words) #이런 벡터가 이미 존재
```

```
## [1] "a"        "able"     "about"    "absolute" "accept"   "account"
```

```
df = tibble(word=words, i=seq_along(words))
head(df)
```

```
## # A tibble: 6 x 2
##   word         i
##   <chr>    <int>
## 1 a            1
## 2 able         2
## 3 about        3
## 4 absolute     4
## 5 accept       5
## 6 account      6
```

```
df %>% filter(str_detect(word, "x$")) #x로 끝나는 단어 세기
```

```
## # A tibble: 4 x 2
##   word      i
##   <chr> <int>
## 1 box     108
## 2 sex     747
## 3 six     772
## 4 tax     841
```

```
str_count(words,"a") #a가 들어간 수 카운트
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 1
##  [38] 1 1 1 1 2 2 2 1 1 2 2 2 1 1 1 2 1 1 1 2 1 1 1 1 3 2 2 1 1 1 1 1 2 1 1 1 1
##  [75] 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0
## [112] 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 2 1 1 1 2 1 1 0 0 0
## [149] 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1
## [186] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1
## [223] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0
## [260] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0
## [297] 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
## [334] 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0
## [371] 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
## [408] 0 0 0 1 1 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [445] 0 0 0 0 0 0 0 1 1 1 1 2 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [482] 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
## [519] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0
## [556] 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0
## [593] 1 1 1 1 1 3 1 1 1 1 2 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0
## [630] 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1
## [667] 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0
## [704] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 2 1 1 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 2
## [741] 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
## [778] 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 2 1 1 1 1 0 0 0 0 0 1 1 0 0
## [815] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
## [852] 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 1 1 1 1
## [889] 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 1 1 1 1
## [926] 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [963] 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
```

```r
str_count("abababa","aba") #카운트는 겹치지않음
```

```
## [1] 2
```

```r
mean(str_count(words,"[aeiou]")) #모음포함율
```

```
## [1] 1.991837
```

```r
str_replace(x, "[aeiou]","-") #교체 근데 하나만 교체됨
```

```
## [1] "-pple"  "b-nana" "p-ar"
```

```r
str_replace_all(x, "[aeiou]","-") #이러면 다 교체
```

```
## [1] "-ppl-"  "b-n-n-" "p--r"
```

## 문자열 추출

```r
head(sentences) #720개의 문장열이다
```

```
## [1] "The birch canoe slid on the smooth planks."
## [2] "Glue the sheet to the dark blue background."
## [3] "It's easy to tell the depth of a well."
## [4] "These days a chicken leg is a rare dish."
## [5] "Rice is often served in round bowls."
## [6] "The juice of lemons makes fine punch."
```

```
colpat=str_c(c("red","orange","yellow","green","blue","purple"),collapse = "|")
colpat
```

```
## [1] "red|orange|yellow|green|blue|purple"
```

```
colsen=str_subset(sentences,colpat) #str_subset은 정규표현식 패턴에 맞는 요소만 남긴다
match=str_extract(colsen,colpat)
match #각 요소에서 패턴에 맞는 문자열을 빼오기
```

```
##  [1] "blue"   "blue"   "red"    "red"    "red"    "blue"   "yellow" "red"
##  [9] "red"    "green"  "red"    "red"    "blue"   "red"    "red"    "red"
## [17] "red"    "blue"   "red"    "blue"   "red"    "green"  "red"    "red"
## [25] "red"    "red"    "red"    "red"    "green"  "red"    "green"  "red"
## [33] "purple" "green"  "red"    "red"    "red"    "red"    "red"    "blue"
## [41] "red"    "blue"   "red"    "red"    "red"    "red"    "green"  "green"
## [49] "green"  "red"    "red"    "yellow" "red"    "orange" "red"    "red"
## [57] "red"
```

```
more = colsen[str_count(colsen, colpat)>1] #응용 색 두개이상만 찾기

more %>% str_extract(colpat)
```

```
## [1] "blue"   "green"  "orange"
```

```
#보면 알겠지만 str_extract는 한 요소에서 처음 찾은거만 리턴한다
more %>% str_extract_all(colpat) #임마는 2차원 리스트로 싹 찾아준다
```

```
## [[1]]
## [1] "blue" "red"
##
## [[2]]
## [1] "green" "red"
##
## [[3]]
## [1] "orange" "red"
```

```
more %>% str_extract_all(colpat, simplify = TRUE) #행렬로 해줌 "간단히"
```

```
##      [,1]     [,2]
## [1,] "blue"   "red"
## [2,] "green"  "red"
## [3,] "orange" "red"
```

```
noun="(a|the) ([^ ]+)" #a the 뒤에 띄어쓰기하고 공백 아닌놈 오는걸로 패턴만듬
has_noun=sentences %>% str_subset(noun) %>% head(10)

has_noun
```

```
##  [1] "The birch canoe slid on the smooth planks."
##  [2] "Glue the sheet to the dark blue background."
##  [3] "It's easy to tell the depth of a well."
##  [4] "These days a chicken leg is a rare dish."
##  [5] "The box was thrown beside the parked truck."
##  [6] "The boy was there when the sun rose."
##  [7] "The source of the huge river is the clear spring."
##  [8] "Kick the ball straight and follow through."
##  [9] "Help the woman get back to her feet."
## [10] "A pot of tea helps to pass the evening."
```

```
has_noun %>% str_extract(noun)
```

```
##  [1] "the smooth" "the sheet"  "the depth"  "a chicken"  "the parked"
##  [6] "the sun"    "the huge"   "the ball"   "the woman"  "a helps"
```

```
has_noun %>% str_match(noun) #요소 각 그룹
```

```
##       [,1]         [,2]  [,3]
##  [1,] "the smooth" "the" "smooth"
##  [2,] "the sheet"  "the" "sheet"
##  [3,] "the depth"  "the" "depth"
##  [4,] "a chicken"  "a"   "chicken"
##  [5,] "the parked" "the" "parked"
##  [6,] "the sun"    "the" "sun"
##  [7,] "the huge"   "the" "huge"
##  [8,] "the ball"   "the" "ball"
##  [9,] "the woman"  "the" "woman"
## [10,] "a helps"    "a"   "helps"
```

```
#tidyr의 extract는 티블에서 작동합니다

tibble(sentence = sentences) %>% extract(sentence, c("article", "noun"), noun, remove = FALSE)
```

```
## # A tibble: 720 x 3
##    sentence                                    article noun
##    <chr>                                       <chr>   <chr>
##  1 The birch canoe slid on the smooth planks.  the     smooth
##  2 Glue the sheet to the dark blue background. the     sheet
##  3 It's easy to tell the depth of a well.      the     depth
##  4 These days a chicken leg is a rare dish.    a       chicken
##  5 Rice is often served in round bowls.        <NA>    <NA>
##  6 The juice of lemons makes fine punch.       <NA>    <NA>
##  7 The box was thrown beside the parked truck. the     parked
##  8 The hogs were fed chopped corn and garbage. <NA>    <NA>
##  9 Four hours of steady work faced us.         <NA>    <NA>
## 10 A large size in stockings is hard to sell.  <NA>    <NA>
## # i 710 more rows
```

```
#결과보면 어떤 원리인지 이해됨, 추출열, 새로운열 이름, 패턴, 기존 데이터없앨지 말지



sentences %>%
  str_replace("([^ ]+) ([^ ]+) ([^ ]+)", "\\1 \\3 \\2") %>%
  head(5)
```

```
## [1] "The canoe birch slid on the smooth planks."
## [2] "Glue sheet the to the dark blue background."
## [3] "It's to easy tell the depth of a well."
## [4] "These a days chicken leg is a rare dish."
## [5] "Rice often is served in round bowls."
```

```
#2번째와 3번째의 표현식의 참조를 통해 자리를 바꿈
```

## 문자열 쪼개기

```
sentences %>% head(5) %>% str_split(" ") #공백기준 스플릿 당연히 정규표현식 가능
```

```
## [[1]]
## [1] "The"     "birch"   "canoe"   "slid"    "on"      "the"     "smooth"
## [8] "planks."
##
## [[2]]
## [1] "Glue"        "the"         "sheet"       "to"          "the"
## [6] "dark"        "blue"        "background."
##
```

```
## [[3]]
## [1] "It's"  "easy"  "to"    "tell"  "the"   "depth" "of"    "a"     "well."
##
## [[4]]
## [1] "These"   "days"    "a"       "chicken" "leg"     "is"      "a"
## [8] "rare"    "dish."
##
## [[5]]
## [1] "Rice"   "is"     "often"  "served" "in"     "round"  "bowls."
```

```
sentences %>% head(5) %>% str_split(" ", simplify = TRUE) #행렬로 간단히
```

```
##      [,1]    [,2]    [,3]    [,4]      [,5]  [,6]    [,7]     [,8]
## [1,] "The"   "birch" "canoe" "slid"    "on"  "the"   "smooth" "planks."
## [2,] "Glue"  "the"   "sheet" "to"      "the" "dark"  "blue"   "background."
## [3,] "It's"  "easy"  "to"    "tell"    "the" "depth" "of"     "a"
## [4,] "These" "days"  "a"     "chicken" "leg" "is"    "a"      "rare"
## [5,] "Rice"  "is"    "often" "served"  "in"  "round" "bowls." ""
##      [,9]
## [1,] ""
## [2,] ""
## [3,] "well."
## [4,] "dish."
## [5,] ""
```

```
dic=c("Name : Hadley", "Country : NZ", "Age : 35") #파이썬의 딕셔너리같은 느낌
dic %>% str_split(" : ", simplify = TRUE, n=2) #이러면 행렬로 변환가능
```

```
##      [,1]      [,2]
## [1,] "Name"    "Hadley"
## [2,] "Country" "NZ"
## [3,] "Age"     "35"
```

```
boundary("word") #복잡하네요
```

```
## [1] NA
## attr(,"options")
## attr(,"options")$type
## [1] "word"
##
## attr(,"options")$skip_word_none
## [1] TRUE
```

```
##
## attr(,"class")
## [1] "stringr_boundary" "stringr_pattern"  "character"
```

```r
sen <- "This is a sentence.  This is another sentence."
str_view_all(sen, boundary("word")) #이런거 가능
```

```
## Warning: `str_view_all()` was deprecated in stringr 1.5.0.
## i Please use `str_view()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## [1] | <This> <is> <a> <sentence>.  <This> <is> <another> <sentence>.
```

```r
str_split(sen, " ")
```

```
## [[1]]
## [1] "This"      "is"        "a"         "sentence." ""          "This"
## [7] "is"        "another"   "sentence."
```

```r
str_split(sen, boundary("word")) #이런것도 가능
```

```
## [[1]]
## [1] "This"     "is"       "a"        "sentence" "This"     "is"       "another"
## [8] "sentence"
```