[R] R에서 사용되는 정규표현식(Regex) 표현 방법과 함수를 통한 사용 예제 — 나무늘보의 개발 블로그

노트북:	blog
만든 날짜:	2020-10-01 오후 8:58
URL:	https://continuous-development.tistory.com/33?category=793392

R

[R] R에서 사용되는 정규표현식(Regex) 표현 방법과 함수를 통한 사용 예제

2020. 7. 22. 08:37 수정 삭제 공개

정규표현식

- 특정한 규칙을 가진 문자열의 집합을 표현하는 데 사용하는 형식 언어

* 0 or more.

\+ 1 or more.

? 0 or 1.

. 무엇이든 한 글자를 의미

^ 시작 문자 지정

ex) ^[abc] abc중 한 단어 포함한 것으로 시작

[^] 해당 문자를 제외한 모든 것 ex) [^abc] a, b, c는 빼고

$ 끝 문자 지정

[a-z] 알파벳 소문자 중 1개

[A-Z] 알파벳 대문자 중 1개

[0-9] 모든 숫자 중 1개

[a-zA-Z] 모든 알파벳 중 1개
[가-힣] 모든 한글 중 1개
[^가-힣] 모든 한글을 제외한 모든 것
[:punct:] 구두점 문자, ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ ] ^ _ ` { | } ~.
[:alpha:] 알파벳 대소문자, 동등한 표현 [A-z]
[:lower:] 영문 소문자, 동등한 표현 [a-z]
[:upper:] 영문 대문자, 동등한 표현 [A-Z].
[:digit:] 숫자, 0,1,2,3,4,5,6,7,8,9,
[:xdigit:] 16진수 [0-9A-Fa-f]
[:alnum:] 알파벳 숫자 문자, 동등한 표현[A-z0-9].
[:cntrl:] ₩n, ₩r 같은 제어문자, 동등한 표현[₩x00-₩x1F₩x7F].
[:graph:] 그래픽 (사람이 읽을 수 있는) 문자, 동등한 표현
[:print:] 출력가능한 문자, 동등한 표현
[:space:] 공백 문자: 탭, 개행문자, 수직탭, 공백, 복귀문자, 서식이송
[:blank:] 간격 문자, 즉 스페이스와 탭.

#grep(pattern,date,[ignore.case],[value]) - 정규표현식을 사용해 원하는 값을 가져오는 함수

pattern - 찾으려는 패턴
date - 데이터
ignore.case - 대소문자 상관없이 찾는다(생략가능)
value - 값을 바로 출력 / 이게 아닐 경우 위치를 출력한다.

```r
##정규표현식 함수()
#grep(pattern, date, ignore.case - 대소문자 상관없이, value - 값을 바로 출력)
?grep

grepValue ← c("gender","name", "age", "hEIght", "wEIght", "tall" , "EIght")
grepValue


#문1) 'ei'로 시작되는 요소(^)가 있는지
grep('^ei',grepValue, ignore.case=T,value=T)

#문2) 'ei'문자열을 포함하는 요소가 있는지
grep('ei',grepValue, value=T)


grepTxt ← c("Bigdata", "Bigdata", "bigdata", "Data", "dataMining", "textMining","campus6","campus5")

grepTxt

#문) b로 시작하는 하나이상의 문자 패턴을 확인하고 싶다면
grep('^b+',grepTxt,value=T)
grep('^b+',grepTxt,ignore.case=T,value=T)
```

```
[1] "Bigdata" "Bigdata" "bigdata"
> grepValue ← c("gender","name", "age", "hEIght", "wEIght", "tall" , "EIght")
> grepValue
[1] "gender" "name"   "age"    "hEIght" "wEIght" "tall"   "EIght"
>
>
> #문1) 'ei'로 시작되는 요소(^)가 있는지
> grep('^ei',grepValue, ignore.case=T,value=T)
[1] "EIght"
>
> #문2) 'ei'문자열을 포함하는 요소가 있는지
> grep('ei',grepValue, value=T)
character(0)
>
>
> grepTxt ← c("Bigdata", "Bigdata", "bigdata", "Data", "dataMining", "textMining","campus6","campus5")
>
> grepTxt
[1] "Bigdata"    "Bigdata"    "bigdata"    "Data"       "dataMining" "textMining" "campus6"    "campus5"
>
```

```
> #문) b로 시작하는 하나이상의 문자 패턴을 확인하고 싶다면
> grep('^b+',grepTxt,value=T)
[1] "bigdata"
> grep('^b+',grepTxt,ignore.case=T,value=T)
[1] "Bigdata" "Bigdata" "bigdata"
```

#gsub - 문자열에서 문자를 바꾸는 기능

```
400  ##gsub( pattern, replacement, data, ignore.case)
401  ##sub
402  #문자열에서 문자를 바꾸는 기능
403
404  # 문) big 이라는 단어를 bigger 라는 단어로 바꾸자고 한다면 ?
405
406  gsub("big","bigger",grepTxt)
407  gsub("big","bigger",grepTxt, ignore.case = T)
408
409  # 문)grepTxt에서 숫자를 제거하고자 한다면?
410  gsub('[0-9]',"",grepTxt)
411  gsub('[[:digit:]]',"",grepTxt)
412
413  sub('[0-9]',"",grepTxt)
414  sub('[[:digit:]]',"",grepTxt)
```

#strsplit(data, split) - 문자열을 기준에 따라 쪼개는 함수

#substr(data, start, stop) - 데이터에서 start/stop까지 의 문자열을 가져오는 함수

```
2  #strsplit(data, split) - 문자열을 쪼개는 함수
3
4  gretingMsg ← "Hi, Bigdata is vary important"
5  strsplit(gretingMsg, " ")
6
7  #substr(data, start, stop) -원하는 길이의 문자열을 가져오는 서브쿼리 함수
8  substr(gretingMsg, 5, 11)
9
0  class(strsplit(gretingMsg, " "))
1
```

```
> #strsplit(data, split) - 문자열을 쪼개는 함수
>
> gretingMsg ← "Hi, Bigdata is vary important"
> strsplit(gretingMsg, " ")
[[1]]
[1] "Hi,"       "Bigdata"   "is"        "vary"      "important"

>
> #substr(data, start, stop) -원하는 길이의 문자열을 가져오는 서브쿼리 함수
> substr(gretingMsg, 5, 11)
[1] "Bigdata"
>
> class(strsplit(gretingMsg, " "))
[1] "list"
```

#str_extract/all - 정규표현식을 통해 추출하는 함수

```
431  #str_extract/all - 정규표현식을 통해 추출하는 함수
432  str_extract("abc123def456","[0-9]{3}")
433  str_extract_all("abc123def456","[0-9]{3}")
434
435  str_extract("abc123def456","[a-z]{3}")
436  str_extract_all("abc123def456","[a-zA-Z]{3}")
437
438  stringDumy ← "임정섭jslim48섭섭해seop34유관순임꺽정홍길동30"
439
440  str_extract_all(stringDumy,"[a-z]{3}") #3자리만 가져온다
441  str_extract_all(stringDumy,"[a-z]{3,}") #최소자리수 3을 넣고 3이상인걸 추출한다.
442  str_extract_all(stringDumy,"[a-z]{3,5}") #최소자리수 3을 넣고 3이상 5이하하인걸 추출한다.
443
```

```
> #str_extract/all - 정규표현식을 통해 추출하는 함수
> str_extract("abc123def456","[0-9]{3}")
[1] "123"
> str_extract_all("abc123def456","[0-9]{3}")
[[1]]
[1] "123" "456"

>
> str_extract("abc123def456","[a-z]{3}")
[1] "abc"
> str_extract_all("abc123def456","[a-zA-Z]{3}")
[[1]]
[1] "abc" "def"

>
> stringDumy ← "임정섭jslim48섭섭해seop34유관순임꺽정홍길동30"
>
> str_extract_all(stringDumy,"[a-z]{3}") #3자리만 가져온다
[[1]]
[1] "jsl" "seo"

> str_extract_all(stringDumy,"[a-z]{3,}") #최소자리수 3을 넣고 3이상인걸 추출한다.
[[1]]
[1] "jslim" "seop"

> str_extract_all(stringDumy,"[a-z]{3,5}") #최소자리수 3을 넣고 3이상 5이하하인걸 추출한다.
[[1]]
[1] "jslim" "seop"
```

예제

```
445
446  #문) 연속된 한글 3자 이상 추출
447  str_extract_all(stringDumy,"[가-힣]{3,}")
448
449  #문) 나이추출
450  str_extract_all(stringDumy,"[0-9]{2}")
451
452  #문) 숫자를 제외
453  str_extract_all(stringDumy,"[^0-9]{3,}")
454
455  ?str_extract_all
456
457  #문) 한글이름 추출(영문자 제외)
458  str_extract_all(stringDumy,"[^a-z]{3,}")
459
```

```
C:/success/R/
> #문) 연속된 한글 3자 이상 추출
> str_extract_all(stringDumy,"[가-힣]{3,}")
[[1]]
[1] "임정섭"            "섭섭해"            "유관순임꺽정홍길동"

>
> #문) 나이추출
> str_extract_all(stringDumy,"[0-9]{2}")
[[1]]
[1] "48" "34" "30"

>
> #문) 숫자를 제외
> str_extract_all(stringDumy,"[^0-9]{3,}")
[[1]]
[1] "임정섭jslim"       "섭섭해seop"        "유관순임꺽정홍길동"

>
> ?str_extract_all
>
> #문) 한글이름 추출(영문자 제외)
> str_extract_all(stringDumy,"[^a-z]{3,}")
[[1]]
[1] "임정섭"               "48섭섭해"             "34유관순임꺽정홍길동30"

>
```

#단어와 숫자에 관련된 메타 문자

```
465
466  ##단어와 숫자에 관련된 메타문자
467  #단어(word) : \\w (영문,한글,숫자,특수문자 모든걸 포함)
468  #숫자(digit): \\d
469  #엔터키,탭키 : \n, \t
470
471  ssn <-"790910-1234567"
472  ssn
473
474  str_extract_all(ssn,"[0-9]{6}-[0-9]{7}")
475  str_extract_all(ssn,"[0-9]{6}-[1-4][0-9]{6}") #여기서  [0-9]{6}이게 묶이고 앞에 제약조건을 [1-4]
476  str_extract_all(ssn,"\\d{6}-[1-4]\\d{6}") # 이런식으로 쓸수도 있다.
477
478
479  email <- "jslim9413@naver.com"
480  email2 <- "jslim9413@naver"
481
482  str_extract_all(email,"\\w{3,}@[a-z]\\w{3,}.[a-z]{2,}")
483  str_extract_all(email2,"\\w{3,}@[a-z]\\w{3,}.[a-z]{2,}")
484
```

```
> ssn <-"790910-1234567"
> ssn
[1] "790910-1234567"
>
> str_extract_all(ssn,"[0-9]{6}-[0-9]{7}")
[[1]]
[1] "790910-1234567"

> str_extract_all(ssn,"[0-9]{6}-[1-4][0-9]{6}") #여기서  [0-9]{6}이게 묶이고 앞에 제약조건을 [1-4]
[[1]]
[1] "790910-1234567"

> str_extract_all(ssn,"\\d{6}-[1-4]\\d{6}") # 이런식으로 쓸수도 있다.
[[1]]
[1] "790910-1234567"

>
>
> email <- "jslim9413@naver.com"
> email2 <- "jslim9413@naver"
>
> str_extract_all(email,"\\w{3,}@[a-z]\\w{3,}.[a-z]{2,}")
[[1]]
[1] "jslim9413@naver.com"

> str_extract_all(email2,"\\w{3,}@[a-z]\\w{3,}.[a-z]{2,}")
[[1]]
character(0)

>
```

'**R**' 카테고리의 다른 글

[R] R로 만드는 제어문 (if, else if, for)과 예제

[R] R에서 사용되는 Data.frame 과 Factor 에 사용되는 다양한 함수

[R] R에 사용되는 배열(array)과 리스트(list)의 개념 및 사용되는 함수

[R] R에 사용되는 행렬(matrix)의 개념 및 사용되는 함수

[R] R에서 사용되는 정규표현식(Regex) 표현 방법과 함수를 통한 사용 예제

[R] R에 사용되는 벡터(matrix)의 개념 및 사용되는 함수(출력,인덱싱,길이반환,문자열비교 등등)

grep gsub R 정규표현식 R 정규표현식 사용법 strsplit str_extract str_extract_all substr 정규표현식

꾸까꾸

혼자 끄적끄적하는 블로그 입니다.