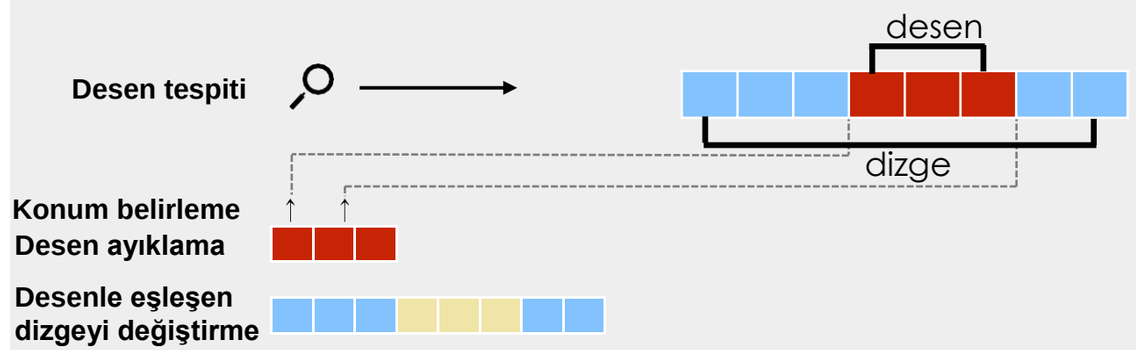
R Dilinde Temel Kurallı İfadeler

HATIRLATICI NOTLAR

Karakter Sınıfları

[[:digit:]] veya \\d	Rakamlar; `[0-9]`
\\D	Rakam olmayan karakterler; `[^0-9]`
[[:lower:]]	Küçük harfler; `[a-z]`
[[:upper:]]	Büyük harfler; `[A-Z]`
[[:alpha:]]	Alfabetik karakterler; `[A-z]`
[[:alnum:]]	Alfanümerik karakterler; `[A-z0-9]`
\\w	Kelime karakterleri; `[A-z0-9_]`
\\W	Kelime karakterleri harici
[[:xdigit:]] veya \\x	Heksadesimal karakterler; `[0-9A-Fa-f]`
[[:blank:]]	Boşluk ve tab
[[:space:]] veya \\s	Boşluk, tab, dikey tab, yeni satır, form besleme, satırbaşı
\\S	Boşluk harici karakter; `[^[:space:]]`
[[:punct:]]	Noktalama karakterleri; `!"#$%&'()*+,-./:;<=>?@[]^_`{|}~`
[[:graph:]]	Grafiksel karakterler; `[[:alnum:][:punct:]]`
[[:print:]]	Yazdırılabilir karakterler; `[[:alnum:][:punct:]\\s]`
[[:cntrl:]] veya \\c	Kontrol karakterleri; `\n, \r` vb.

Desen Eşleme Fonksiyonları




```
> dizge <- c("kedi", "aslan", "kaplan")
> desen <- "l.n"
```

Desen Tespiti

grep(desen, dizge)

```
[1] 2 3
```

grep(desen, dizge, value = TRUE)

```
[1] "aslan"
[2] "kaplan"
```

grepl(desen, dizge)

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

stringr::str_detect(dizge, desen)

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

Desenle Eşleşen Dizgeyi Bölme

strsplit(dizge, desen) veya stringr::**str_split(dizge, desen)**

Eşleşen Desenin Konumunu Bulma

regexpr(desen, dizge)

İlk eşleşmenin başlangıç pozisyonunu ve eşlemenin uzunluğunu bulur.

gregexpr(desen, dizge)
Tüm eşleşmelerin başlangıç ve bitiş pozisyonunu bulur.

stringr::**str_locate(dizge, desen)**
İlk eşleşmenin başlangıç ve bitiş pozisyonunu bulur.

stringr::**str_locate_all(dizge, desen)**
Tüm eşleşmelerin başlangıç ve bitiş pozisyonunu bulur.

Desenle Eşleşen Dizgeyi Ayıklamak

regmatches(dizge, regexpr(desen, dizge))
İlk eşleşmeyi ayıklar. `[1] "lan" "lan"`

regmatches(dizge, gregexpr(desen, dizge))
Tüm eşleşmeleri bir *list* yapısında ayıklar.
`[[1]] character(0) [[2]] "lan" [[3]] "lan"`

stringr::**str_extract(dizge, desen)**
İlk eşleşmeyi ayıklar. `[1] NA "lan" "lan"`

stringr::**str_extract_all(dizge, desen)**
Tüm eşleşmeleri bir *list* yapısında ayıklar.

stringr::**str_extract_all(dizge, desen, simplify = TRUE)**
Tüm eşleşmeleri bir *matrix* yapısında ayıklar.

stringr::**str_match(dizge, desen)**
İlk eşleşmenin tamamını ve eşleşen her bir karakter grubunu *matrix* yapısında ayıklar.

stringr::**str_match_all(dizge, desen)**
Tüm eşleşmelerin tamamını ve eşleşen her bir karakter grubunu *matrix* yapısında ayıklar.

Desenle Eşleşen Dizgeyi Değiştirme

sub(desen, yenideger, dizge)
Desene uyan ilk eşleşmeyi değiştirir.

gsub(desen, replacement, dizge)
Tüm eşleşmeleri değiştirir.

stringr::**str_replace(dizge, desen, replacement)**
Desene uyan ilk eşleşmeyi değiştirir.

stringr::**str_replace_all(dizge, desen, replacement)**
Desene uyan tüm eşleşmeleri değiştirir.

Özel Metakarakterler

\n	Yeni satır
\r	Satır başı
\t	Tab
\v	Dikey tab
\f	Form besleme

İki Yönlü Eşleme ve Koşullar*

(?=)	İleriye dönük eşleme (PERL = TRUE iken), Örn:`(?=yx)`: Mevcut konumdan sonra 'xy' olmalı.
(?!)	Olumsuz ileriye dönük eşleme (PERL = TRUE iken); mevcut konumdan sonra bu desen gelmemeli.
(?<=)	Geriye dönük eşleme (PERL = TRUE iken), Örn: `(?<=yx)`: mevcut konumdan sonra `'xy'` gelmeli.
(?<!)	Olumsuz geriye dönük eşleme (PERL = TRUE); mevcut konumdan sonra bu desen olmamalı.
?(if)then	If-then-koşulu (PERL = TRUE iken); ileriye dönük eşlemede kullanılır.
?(if)then|else	If-then-else-koşulu (PERL = TRUE iken)

*bknz., örn: http://www.regular-expressions.info/lookaround.html
http://www.regular-expressions.info/conditional.html

Karakter Sınıflama ve Gruplama

.	`\n` harici herhangi bir karakter
|	VEYA. Örneğin `(a|b)`
[…]	İzinli karakter listesi örn:`[abc]`
[a-z]	Karakter aralığı belirtir
[^…]	Karakterleri hariç tutar
(…)	Gruplama ve `\\N` kere geri referans verme (N: tam sayı)

Çapalar

^	Dizgenin başlangıcı
$	Dizgenin bitişi
\\b	Bir kelimenin başı veya sonundaki boş dizge
\\B	Bir kelimenin başı veya sonu olmamalı
\\<	Kelime başı
\\>	Kelime sonu

Miktar Belirleyiciler

*	En az 0 kere eşler
+	En az 1 kere eşler
?	En fazla 1 kere eşler
{n}	Tam olarak n kere eşler
{n,}	En az n kere eşler
{,n}	En fazla n kere eşler
{n,m}	En az n, en fazla m kere eşler

Genel Modlar

R varsayılan olarak *POSIX extended* kurallı ifadeleri kullanır. *PCRE* kurallı ifadeleri kullanmak için `PERL = TRUE` vermek veya deseni `perl()` içinde vermek gerekir.
Tüm temel fonksiyonlar `fixed = TRUE` parametresi ile kullanılırsa veya stringr paketinde `fixed` fonksiyonu kullanılırsa verilen desen metin olarak değerlendirilir ve desen eşlemesi yapılmaz, birebir metin araması yapılır.
Temel fonksiyonların tamamı `ignore.cases = TRUE` parametresi ile büyük-küçük harf duyarlı hale getirilebilir.

Kaçış Karakterleri

Aranılacak dizge içerisinde `* + ?` gibi desen karakterleri yer alıyorsa bu karakterlerin metakarakter olarak algılanmaması (çakışmayı önlemek) için ters bölü `\\` kullanılabilir veya o karakter `\\Q` ile `\\E` arasında yazılabilir.

Büyük-Küçük Harf Duyarlılığı

Kurallı ifadelerde `(?i)` ile büyük-küçük harf duyarsız arama yaptırılabilir. Geri aramalarda dizge `\\L` ile küçük harfe `\\U` ile büyük harfe çevrilebilir (Örn. `\\L\\1`). `PERL = TRUE` iken çalışır.

Açgözlü Eşleme

***** metakarakteri varsayılan olarak açgözlüdür, yani mümkün olduğu kadar uzun dizgeyle eşleme yapmak ister. Tembel eşleme modunda `?` metakarakteri ile de kullanılabilir. Örn:`*?`
Açgözlü eşleme modunu kapatmak veya kapalıysa aktif hale getirmek için `(?U)`kullanılır. Örneğin `(?U)a*` tembel eşleme, `(?U)a*?` açgözlü eşleme yapar.

Hatırlatma

Kurallı ifadeler rex::**rex()** fonksiyonu yardımıyla da oluşturulabilir.

Çeviri: Dr. Zeki Özen • zekiozen@istanbul.edu.tr • Çeviri Tarihi: 01/18