

R Dilinde Temel Kurallı İfadeler

HATIRLATICI NOTLAR

Karakter Sınıfları

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

Özel Metakarakterler

<code>\n</code>	Yeni satır
<code>\r</code>	Satır başı
<code>\t</code>	Tab
<code>\v</code>	Dikey tab
<code>\f</code>	Form besleme

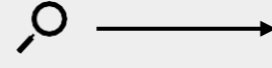
İki Yönlü Eşleme ve Koşullar*

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

*bnkz., örn: <http://www.regular-expressions.info/lookaround.html>
<http://www.regular-expressions.info/conditional.html>

Desen Eşleme Fonksiyonları

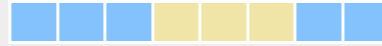
Desen tespiti



Konum belirleme
Desen ayıklama



Desenle eşleşen dizgeyi değiştirme



```
> dizge <- c("kedi", "aslan", "kaplan")
> desen <- "1.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

`regexr(desen, dizge)`

İlk eşleşmenin başlangıç pozisyonunu ve eşleşmenin 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, regexr(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.

Karakter Sınıflama ve Grublama

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

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.

Çapalar

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

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 `regex::rex()` fonksiyonu yardımıyla da oluşturulabilir.