

## Expressões Regulares com o R e processamento de texto

```
> rm(list=ls())
```

Anotação 1 - Comando rm

```
> require(stringr)
```

Anotação 2 - Comando require

Anotação 3 - Biblioteca stringr

```
> #Exemplo de datas
```

```
> string = "23 mai 2000"
```

```
> string
```

```
[1] "23 mai 2000"
```

```
> string2 = "1 mai 2000"
```

```
> string2
```

```
[1] "1 mai 2000"
```

```
> string3 = "1 maio 2000"
```

```
> string3
```

```
[1] "1 maio 2000"
```

```
> regexp <- "([[:digit:]]{2}) ([[:alpha:]]+) ([[:digit:]]{4})"
```

```
> regexp
```

```
[1] "([[:digit:]]{2}) ([[:alpha:]]+) ([[:digit:]]{4})"
```

Anotação 4 - O que a expressão regular descreve?

Comandos nativos

```
> #Nativo
```

```
> grepl(pattern = regexp, x = string)
```

```
[1] TRUE
```

```
> grepl(pattern = regexp, x = string2)
```

```
[1] FALSE
```

```
> grepl(pattern = regexp, x = string3)
```

```
[1] FALSE
```

#### Anotação 5 - Anotações

```
> #Pacote stringr
```

```
> str_detect(string, regexp)
```

```
[1] TRUE
```

```
> str_detect(string, regexp)
```

```
[1] TRUE
```

```
> str_detect(string, regexp)
```

```
[1] TRUE
```

#### Anotação 6 - Anotações

```
> string4 = "10 cas 2000"
```

```
> grepl(pattern = regexp, x = string4)
```

```
[1] TRUE
```

#### Anotação 7 - Anotações

#### Anotação 8 - Como deve ser a expressao regular para validar uma data no formato proposto?

```
> data1 = "08/10/2014"
```

```
> data2 <- "8/10/2014"
```

Anotação 9 - Como deve ser a expressao regular para validar essas datas? Crie o padrão e o teste no R

```
> telefone1 = "99821-8070"
> telefone2 = "9906-8170"
```

Anotação 10 - Qual a expressão para detectar um telefone com 8 e 9 dígitos? Crie o padrão e o teste no R

```
> #Contando a ocorrência de padrões em uma string
> string <- "blabla 23 mai 2000 blabla 18 mai 2004 a aa c"
> string
```

```
[1] "blabla 23 mai 2000 blabla 18 mai 2004 a aa c"
```

```
> require(tau)
> textcnt(string,n=1L,method="string")
```

```
      a      aa blabla      c      mai
      1      1      2      1      2
attr(,"useBytes")
[1] FALSE
attr(,"class")
[1] "textcnt"
```

```
> textcnt(string,n=5L,method="string")
```

```
      blabla mai a aa c blabla mai blabla mai a      mai blabla mai a aa
      1              1              1
attr(,"useBytes")
[1] FALSE
attr(,"class")
[1] "textcnt"
```

Anotação 11 - Anotações

```
> #Extraindo a posicao de uma substring
> texto <- letters
> texto
```

```

[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"

> grep(pattern = "p",x = texto )

[1] 16

> grep(pattern = "[a-z]",x = texto)

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26

```

#### Anotação 12 - Anotações

```

> #Extraindo a posicao de um padrao
> regexp <- "([[:digit:]]{2}) ([[:alpha:]]+) ([[:digit:]]{4})"
> string <- "blabla 23 mai 2000 blabla 18 mai 2004"
> regexpr(pattern = regexp, text = string)

[1] 8
attr(,"match.length")
[1] 11
attr(,"useBytes")
[1] TRUE

> gregexpr(pattern = regexp, text = string)

[[1]]
[1] 8 27
attr(,"match.length")
[1] 11 11
attr(,"useBytes")
[1] TRUE

> str_locate(string,regexp)

      start end
[1,]      8 18

> str_locate_all(string,regexp)

[[1]]
      start end
[1,]      8 18
[2,]     27 37

```

#### Anotação 13 - Anotações

```
> #Extrair trechos de uma string
> substr("simple text",1,3)

[1] "sim"

> str_sub("simple text",2,4)

[1] "imp"
```

#### Anotação 14 - Anotações

```
> #Extraindo padroes em uma string
> regexp <- "([[:digit:]]{2}) ([[:alpha:]]+) ([[:digit:]]{4})"
> string <- "blabla 23 mai 2000 blabla 18 mai 2004"
> str_extract(string,regexp)

[1] "23 mai 2000"

> str_extract_all(string,regexp)

[[1]]
[1] "23 mai 2000" "18 mai 2004"

> str_match(string,regexp)

      [,1]      [,2] [,3] [,4]
[1,] "23 mai 2000" "23" "mai" "2000"

> str_match_all(string,regexp)

[[1]]
      [,1]      [,2] [,3] [,4]
[1,] "23 mai 2000" "23" "mai" "2000"
[2,] "18 mai 2004" "18" "mai" "2004"
```

#### Anotação 15 - Anotações

```
> #Substituindo em uma string
> string <- "23 mai 2000"
> regexp <- "([[:digit:]]{2}) ([[:alpha:]]+) ([[:digit:]]{4})"
> sub(pattern = regexp, replacement = "\\1", x = string) #retrovisor primeira parte

[1] "23"

> sub(pattern = regexp, replacement = "\\2", x = string) #retrovisor segunda parte
```

```
[1] "mai"
```

```
> sub(pattern = regexp, replacement = "\\3", x = string) #retrovisor terceira parte
```

```
[1] "2000"
```

#### Anotação 16 - Anotações

```
> text <- "abc def ghk"
```

```
> sub(pattern = " ", replacement = "", x = text)
```

```
[1] "abcdef ghk"
```

```
> gsub(pattern = " ", replacement = "", x = text)
```

```
[1] "abcdefghk"
```

#### Anotação 17 - Anotações

```
> chartr(old="a",new="o",x="baba")
```

```
[1] "bobo"
```

```
> chartr(old="ab",new="ot",x="baba")
```

```
[1] "toto"
```

```
> str_replace_all("abc.def.ghi.jkl", "\\.", "_")
```

```
[1] "abc_def_ghi_jkl"
```

#### Anotação 18 - Anotações

```
> #Manipulando uma string
```

```
> str_pad("abc",width=10,side="both",pad = "+")
```

```
[1] "+++abc++"
```

```
> str_pad(c("1","11","111","1111"),3,side="left",pad="0")
```

```
[1] "001" "011" "111" "1111"
```

#### Anotação 19 - Anotações

```
> #Removendo espacos em branco
> #install.packages("memisc", repos="http://R-Forge.R-project.org")
> library("memisc")
> trimws(" abc def ")

[1] "abc def"

> library("gdata")
> trim(" abc def ")

[1] "abc def"

> str_trim(" abd def ")

[1] "abd def"
```

#### Anotação 20 - Anotações

## 1 Referências

#### Anotação 21 - Anotações