R para Ciência de Dados 2

Stringr

agosto de 2021

Motivação

Bases com colunas em texto já são *extremamente* comuns hoje em dia, então saber lidar com strings se torna essencial na caixa de ferramentas do cientista de dados

Além de ajudar em análise de dados, tratar strings ajuda com programação porque grande parte das linguagens modernas funcionam da mesma maneira que o R nesse quesito

O conhecimento de expressões regulares vale para a vida, é impossível descrever com poucas palavras todas as coisas que são implementáveis via regex

Normalmente os textos são bagunçados, independentemente do quão cuidadosa foi a coleta de dados, então precisamos arrumá-los; podemos fazer isso do jeito fácil (`{stringr}` e regex) ou do jeito difícil (`{base}` e lágrimas)

Introdução

- Strings não passam sequências de caracteres ("cadeias" em português)
- No R podemos criar uma string com um par de aspas (simples ou duplas)
- O `print()` mostra a estrutura da string, enquanto `cat()` mostra o texto

```
print("こんにちは! Está 10\u00BAC na lá fora")
```

```
#> [1] "こんにちは! Está 10ºC na lá fora"
```

- Para colocar aspas dentro de uma string, podemos **escapar** o caractere

```
cat("Ele disse \"escapar\"")
```

```
#> Ele disse "escapar"
```

O pacote {stringr}

- O pacote `{stringr}` é a forma mais simples de trabalhar com strings no R

```
library(stringr)

abc <- c("a", "b", "c")
str_c("prefixo-", abc, "-sufixo")
```

```
#> [1] "prefixo-a-sufixo" "prefixo-b-sufixo" "prefixo-c-sufixo"
```

- Todas as funções relevantes começam com `str_` e funcionam bem juntas

```
abc %>%
  str_c("-sufixo") %>%
  str_length()
```

```
#> [1] 8 8 8
```

Principais funções

Função(ões)	Significado
`str_c`	Colar strings
`str_length`	Contagem de caracteres na string
`str_detect`	O padrão existe na string?
`str_extract[_all]`	Extrair o padrão da string
`str_replace[_all]`	Substituir um padrão por outro na string
`str_remove[_all]`	Remover um padrão da string
`str_split`	Quebrar a string em pedaços
`str_squish`	Remover espaços extras da string
`str_sub`	Extrair um pedaço da string
`str_to_[lower/upper]`	Converter a string para caixa baixa/alta
`str_to_[sentence/title]`	Converter no formato de frase ou título

Exemplos

```
str_detect("Colando Strings", pattern = "ando")
```

```
#> [1] TRUE
```

```
str_extract("Colando Strings", pattern = "ando")
```

```
#> [1] "ando"
```

```
str_replace("Colando Strings", pattern = "ando", replacement = "ei")
```

```
#> [1] "Colei Strings"
```

```
str_remove("Colando Strings", pattern = " Strings")
```

```
#> [1] "Colando"
```

Exemplos (cont.)

```
str_split("Colando Strings", pattern = " ")
```

```
#> [[1]]
#> [1] "Colando" "Strings"
```

```
str_squish("   Colando   Strings   ")
```

```
#> [1] "Colando Strings"
```

```
str_sub("Colando Strings", start = 1, end = 7)
```

```
#> [1] "Colando"
```

```
str_to_lower("Colando Strings")
```

```
#> [1] "colando strings"
```

Regex

- **Expressões regulares** são "programação para strings", permitindo extrair padrões bastante complexos com comandos simples
- Elas giram em torno de padrões "normais" de texto, mas com alguns símbolos especiais com significados específicos

```
frutas <- c("banana", "TANGERINA", "maçã", "lima")
str_detect(frutas, pattern = "na")
```

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

- Exemplos: `.` (qualquer caractere), `^` (início da string) e `$` (fim da string)

```
str_detect(frutas, pattern = "^ma")
```

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

Mais regex

- Podemos contar as ocorrências de um padrão: `+` (1 ou mais vezes), `*` (0 ou mais vezes), `{m,n}` (entre `m` e `n` vezes), `?` (0 ou 1 vez)

```
ois <- c("oi", "oii", "oiii!", "oioioi!")
str_extract(ois, pattern = "i+")
```

```
#> [1] "i"   "ii"  "iii" "i"
```

- `[]` é um conjunto e `()` é um conjunto "inquebrável"

```
str_extract(ois, pattern = "[i!]$")
```

```
#> [1] "i" "i" "!" "!"
```

```
str_extract(ois, pattern = "(oi)+")
```

```
#> [1] "oi"     "oi"     "oi"     "oioioi"
```

Ainda mais regex

- Se de fato precisarmos encontrar um dos **caracteres reservados** descritos anteriormente, precisamos escapá-los da mesma forma como vimos antes

```
str_replace("Bom dia.", pattern = ".", replacement = "!")
```

```
#> [1] "!om dia."
```

```
str_replace("Bom dia.", pattern = "\\.", replacement = "!")
```

```
#> [1] "Bom dia!"
```

- Não esquecer que algumas funções do `{stringr}` possuem variações

```
str_replace_all("Bom. Dia.", pattern = "\\.", replacement = "!")
```

```
#> [1] "Bom! Dia!"
```

Exemplos intermináveis

```
str_subset(c("banana", "TANGERINA", "maçã", "lima"), "NA")  # Maiúscula
```

```
#> [1] "TANGERINA"
```

```
str_subset(c("banana", "TANGERINA", "maçã", "lima"), "^ma") # Início
```

```
#> [1] "maçã"
```

```
str_subset(c("banana", "TANGERINA", "maçã", "lima"), "ma$") # Final
```

```
#> [1] "lima"
```

```
str_subset(c("banana", "TANGERINA", "maçã", "lima"), ".m")  # Qualquer
```

```
#> [1] "lima"
```

Exemplos intermináveis (cont.)

```
str_extract(c("oii", "oiii!", "oiii!!!", "oioioi!"), "i+!")    # 1 ou mais
```

```
#> [1] NA     "iii!" "iii!" "i!"
```

```
str_extract(c("oii", "oiii!", "oiii!!!", "oioioi!"), "i+!?")   # 0 ou 1
```

```
#> [1] "ii"   "iii!" "iii!" "i"
```

```
str_extract(c("oii", "oiii!", "oiii!!!", "oioioi!"), "i+!*")   # 0 ou mais
```

```
#> [1] "ii"     "iii!"   "iii!!!" "i"
```

```
str_extract(c("oii", "oiii!", "oiii!!!", "oioioi!"), "i{1,2}") # Entre m e n
```

```
#> [1] "ii" "ii" "ii" "i"
```

Exemplos intermináveis (cont.)

```
str_extract(c("oii", "oiii!", "oiii!!!", "oioioi!"), "[i!]+")  # Algum
```

```
#> [1] "ii"     "iii!"   "iii!!!" "i"
```

```
str_extract(c("banana", "TANGERINA", "maçã", "lima"), "[a-z]") # Conjuntos
```

```
#> [1] "b" NA  "m" "l"
```

```
str_extract(c("oii", "oiii!", "oiii!!!", "oioioi!"), "(oi)+")  # Tudo
```

```
#> [1] "oi"     "oi"     "oi"     "oioioi"
```

```
str_extract(c("oii", "oiii!", "ola!!!", "oioioi!"), "(i+|!+)") # Ou
```

```
#> [1] "ii"  "iii" "!!!" "i"
```

Exemplos intermináveis (cont.)

```
str_replace("Bom dia.", "\\.", "!")      # Escapando
```

```
#> [1] "Bom dia!"
```

```
str_replace("Bom. Dia.", "\\.", "!")     # Primeira ocorrência
```

```
#> [1] "Bom! Dia."
```

```
str_replace_all("Bom. Dia.", "\\.", "!") # Lembrar do _all
```

```
#> [1] "Bom! Dia!"
```

```
str_remove_all("Bom \"dia\"", "\\\"")    # Escapando escape
```

```
#> [1] "Bom dia"
```

Exemplos intermináveis (cont.)

```
stringi::stri_trans_general("Váríös àçêntõs", "Latin-ASCII") # Remover acentos
```

```
#> [1] "Varios acentos"
```

```
str_extract_all("Número: (11) 91234-1234", "[0-9]+")          # Números
```

```
#> [[1]]
#> [1] "11"    "91234" "1234"
```

```
str_extract("Número: (11) 91234-1234", "[A-Za-z]+")          # Conjuntos juntos
```

```
#> [1] "N"
```

```
str_extract("Número: (11) 91234-1234", "[:alpha:]+")          # Acentos
```

```
#> [1] "Número"
```