Scraping (força bruta)

Igor F. Nascimento

17 de outubro de 2017

Scraping

Introdução

Data scraping (do inglês, raspagem de dados) é uma técnica computacional na qual um programa extrai dados de saída legível somente para humanos, proveniente de um serviço ou aplicativo.

Sites

- Transfermarkt
- FIFA
- CS

Pacotes

```
library(RCurl)
library(data.table)
library(gsubfn)
library(stringr)
library(rvest)
```

Funções

```
gsub
grep
getURL
strapplyc
unlist
strsplit
```

Expressão regular

Uma expressão regular é uma notação para representar padrões em strings. Serve para validar entradas de dados ou fazer busca e extração de informações em textos.

Expressão regular - sintaxe

- \\d: dígito (0,1,...9)
- \\D: não dígito
- \\w: letra
- \\W: não letra
- \\s: espaço
- \\S: não espaço
- ^: início do texto
- $: fim do texto
- .: qualquer caracter
- \[ab \]: conjunto a ou b
- \[^ab \]: exceto a ou b
- \[A-Z \]: letras maiúsculas
- i+: i pelo menos 1 vez
- i*: i zero ou mais vezes
- i?: i zero ou 1 vez
- i{n}: i n vezes
- i{n,}: i pelo menos 1 vez
- i{n,m}: i pelo menos n e não mais que m

Expressão regular - exercício

- cpf: 123.456.789-00
- telefone: (61) 98765-4321
- telefone: (61) 8765-4321
- telefone geral: com e sem 9

grep

Procura e retorna a posição (vetor) ou objeto em um padrão a ser procurado em um texto.

- **pattern**: padrão procurado
- **x**: objeto com os valores a serem procurados e substituídos
- **value**:

grep entrada fixa

```
texto <- c("eu amo R","eu amo LAMFO","eu odeio SVM")
grep("eu amo",texto)
```

```
## [1] 1 2
```

```
grep("eu amo",texto,value = T)
```

```
## [1] "eu amo R"      "eu amo LAMFO"
```

grep entrada variavel

```
texto <- c("eu amo R","eu amo LAMFO","eu odeio SVM","eu det
grep("eu\\s(amo|odeio)",texto,value=T)
```

```
## [1] "eu amo R"      "eu amo LAMFO" "eu odeio SVM"
```

```
grep("eu\\s\\w*\\s\\w*",texto,value = T)
```

```
## [1] "eu amo R"                  "eu amo LAMFO"
## [3] "eu odeio SVM"              "eu detesto estatística"
```

gsub

A função *gsub* substitui um determinado **padrão** de texto por outro **padrão** em um texto ou vetor.

- **pattern**: padrão procurado
- **replacement**: padrão substituto
- **x**: objeto com os valores a serem procurados e substituídos
- **ignore.case**: *case sensitive*

gsub - entrada fixa

```
texto <- "eu amo python"
gsub("Python","R",texto,ignore.case = T)
```

```
## [1] "eu amo R"
```

gsub - entrada variável

- tudo entre parêntes permanece

```
texto <- c("eu amo R","eu amo LAMFO","eu odeio SVM","eu det
gsub("(eu amo ).*","\\1R",texto,ignore.case = T)
```

```
## [1] "eu amo R"                  "eu amo R"
## [3] "eu odeio SVM"              "eu detesto estatística"
```

gsub - entrada variável multivariada

```
texto <- "eu amo SAS e não gosto de R"
gsub("(eu amo )[a-z]*( e não gosto de )[a-z]*","\\1R\\2SAS"
```

```
## [1] "eu amo R e não gosto de SAS"
```

gsub - desafio

Transforme o nome “Pedro Henrique de Melo Albuquerque” em padrão de citação.

```
## [1] "Albuquerque, P. H. d. M."
```

strapplyc

Encontra todos os casos com um padrão em um texto

- **X**: objeto com os valores a serem procurados e substituídos
- **pattern**: padrão procurado
- **simplify**: "c" simplifica o objeto para um único vetor
- **ignore.case**: *case sensitive*

strapplyc exemplo 1

```
library(gsubfn)
```

```
## Loading required package: proto
```

```
strapply(c("12;34:56","12,23,34.54:56,89,,12","12.12.12"),
```

```
## [[1]]
## [1] "12" "34" "56"
##
## [[2]]
## [1] "12" "23" "34" "54" "56" "89" "12"
##
## [[3]]
## [1] "12" "12" "12"
```

```
strapply(c("12;34:56","12,23,34.54:56,89,,12","12.12.12"),
```

```
##  [1] "12" "34" "56" "12" "23" "34" "54" "56" "89" "12"
```

strsplit

Divide todos os casos entre um padrão em um texto

- **X**: objeto com os valores a serem procurados e substituídos
- **split**: padrão procurado
- **simplify**: “c” simplifica o objeto para um único vetor
- **ignore.case**: *case sensitive*

strsplit exempplo 1

```
strsplit(c("12.34.56"), "\\.")
```

```
## [[1]]
## [1] "12" "34" "56"
```

getURL

Lendo código fonte do html de um página

- url: link página
- ssl.verifyhost=FALSE
- ssl.verifypeer=FALSE

Função “url.exists” antes

Retora 1 objeto caracter

getURL exemplo

```
link <- "https://www.fifaindex.com/pt-br/"
html <- getURL(link,
               ssl.verifyhost=FALSE, ssl.verifypeer=FA
html <- unlist(strsplit(html,"\\n"))
```

read_html

Lendo código fonte do html de um página

- x: link página

Função “url.exists” antes

Retora um vetor (linha a linha) de caracteres

read_html exemplo

```
link <- "https://www.transfermarkt.com/premier-league/s
html <- read_html(link, encoding = "UTF-8")
html <- htmlParse(html)
html <- capture.output(html)
```