Web scraping com R

Clube de Ciência de Dados

Universidade Federal Fluminense

CCD-UFF

Clube de Ciência de Dados

Conteúdo

- O que é *'Web scraping'?*
  - Usos de *Web scraping*
  - Técnicas de *Web scraping*
- Como realizar *Web scraping* com a linguagem R
- Caso de estudo: 'ondefuiroubado.com.br'

O que é *'Web scraping'?*

O que é *'Web scraping'?*

- É uma técnica de extração de informação à partir de sites da *web*
- Está relacionado com *'Web indexing'*, o qual indexa informação da *web* usando *'bots'* ou *'web crawlers'*
- Foca na transformação de dado não-estruturado na *web* (geralmente em formato *HTML*) em dado estruturado, que pode ser guardado e analisado

Usos de *Web scraping*

- Usos incluem:
  - Comparações de preços online;
  - *'Contact scraping'* (prática de obtenção de endereços de *email*, geralmente usado para propósitos de *marketing*);
  - Monitoramento de dados meteorológicos,
  - Detecção de mudanças em *websites*;
  - Pesquisa;
  - *'Web Mashup'* (página ou aplicação *web* que usa conteúdo de mais de uma fonte para criar um único serviço a ser exibido);
  - Integração de dados da *web*

Técnicas de *Web scraping*

- *Human copy-and-paste*
- *Text grepping and regular expression matching*
- *Http programming*
- *Html parsers*
- *DOM parsing*
- *Web-scraping software*
- *Vertical aggregation platforms*
- *Semantic annotation recognizing*
- *Computer Vision web-page analyzers*

Técnicas de *Web scraping*: *Human copy-and-paste*

- Examinação manual realizada por um humano.
- As vezes é a única solução possível. Por exemplo, quando os sites explicitamente colocam barreiras para prevenir *scraping* automatizado pelo computador

Técnicas de *Web scraping: Text grepping and regular expression matching*

- Uma abordagem simples e poderosa para extrair informação.
- Baseado no comando do *UNIX*, *'grep'*, ou no uso de expressões regulares
- Presente em linguagens como *Perl*, *Python* e *R*

Técnicas de *Web scraping: Html parsers*

- Muitos *websites* tem grandes coleções de páginas geradas dinamicamente a partir de uma fonte, como um banco de dados, por exemplo
- Dados de uma mesma categoria são tipicamente codificados em páginas similares por um *script* ou *template* em comum
- Algumas linguagens de consulta semi-estruturada, como o *Xquery*, *HTQL* e *XPath* podem ser usadas para analisar páginas *HTML*, recuperar e transformar conteúdo da página

Técnicas de *Web scraping: DOM parsing*

- Semelhante ao *Html parser*
- Pode recuperar conteúdo gerado dinamicamente por *scripts* na página

Técnicas de *Web scraping: Semantic annotation recognizing*

- Análise de metadata ou de anotações e marcações semânticas
- Pode ser usada para identificar partes específicas do *site* antes da recuperação de informação propriamente dita

Técnicas de *Web scraping: Computer Vision web-page analyzers*

- Uso de Aprendizado de Máquina e Computação Visual para identificar e extrair informação através de interpretação visual da página
- Simula o comportamento usual de uma pessoa visualizando a página

Como realizar *Web scraping* com a linguagem R

Como realizar *Web scraping* com a linguagem R – *Html Parser*

- `htmlTreeParse()` presente no pacote 'XML', (`library(XML)`), recupera toda a página html
  - Arguments:
    - `file =` `# url da página do site`
    - `useInternalNodes = TRUE`
    - `encoding = "UTF-8"`
- `xpathSApply()` também presente no pacote 'XML', recupera informação de tags
  - Arguments:
    - `doc =` `# variável que está guardando a página html`
    - `path =` `# (xpath expression)`
    - `namespaces = xmlValue`

Como realizar *Web scraping* com a linguagem R – *Html Parser*

- *XPath Syntax*:
  - http://www.w3schools.com/xpath/xpath_syntax.asp
  - Torna fácil a seleção de *tags* específicas e a obtenção da informação desejada
  - Mais detalhes do uso *XPath* (exemplos) na seção de caso de estudo

Caso de estudo: ‘ondefuiroubado.com.br’

Caso de estudo: ‘ondefuiroubado.com.br’

- O ‘www.ondefuiroubado.com.br’ é um site onde pessoas voluntariamente notificam que foram assaltadas ou roubadas
- Algumas informações disponíveis na página *html*:
  - Localização (latitude, longitude e cidade)
  - Tipo de ocorrência
  - Objetos roubados
  - Data e Hora
  - Título e descrição da ocorrência
- Exemplo: http://www.ondefuiroubado.com.br/denuncias/1

Caso de estudo: ‘ondefuiroubado.com.br’

- No *github* está listado o script utilizado para realizar o *Web scraping* da página (código necessita de testes e revisões mais abrangentes)
  - https://github.com/ClubedeCienciadeDadosUFF/CCDrepository

Caso de estudo: 'ondefuiroubado.com.br'

- A estratégia básica consiste em, para cada página:
  - 1. Passar a *URL* (http://www.ondefuiroubado.com.br/denuncias/'numero_da_ocorrência' , onde o 'numero_da_ocorrência' é um número , entre 1 e 50000, por exemplo) para a função `htmlTreeParse()` (como descrito na seção anterior);
  - 2. Recuperar as informações da página com o *Xpath*;
  - 3. Organizar as informações em um *dataset*

Caso de estudo: ‘ondefuiroubado.com.br’ -1-

```
BASE_URL <- "http://www.ondefuiroubado.com.br/denuncias/"

 url <- paste(BASE_URL, n, sep = "")  # n é um valor a ser modificado em cada iteração e representa o numero da ocorrência no site

html <- htmlTreeParse(url, useInternalNodes = TRUE, encoding = "UTF-8")
```

Caso de estudo: ‘ondefuiroubado.com.br’ -2-

```
n_occurrence <- xpathSApply(html, "//*[@id=\"id_report\"]",
xmlValue)

…

latitude                "//*[@id=\"lat\"]"

longitude               "//*[@id=\"lng\"]"

city                    "//*[@class=\"hc-city-name\"]"

occurrence_type         "//*[@class=\"sd-info-type\"]"

occurrence_title        "//*[@class=\"sd-info-title\"]"

spoil                   "//*[@class=\"obj-label valign-middle\"]"

date_time               "//*[@class=\"sd-info-data-hora\"]"

occurrence_description  "//*[@class=\"description valign-top\"]"
```

Caso de estudo: ‘ondefuiroubado.com.br’ -2-

“view-source:http://www.ondefuiroubado.com.br/denuncias/1” – CTRL + F

"//*[@id=\"id_report\"]“

** 1 **

"//*[@id=\"lat\"]“

** -12.9777431 **

"//*[@id=\"lng\"]“

** -38.4312059 **

"//*[@class=\"hc-city-name\"]“

<h4 class="hc-city-name">Salvador </h4>

Caso de estudo: ‘ondefuiroubado.com.br’ -2-

"//*[@class=\"sd-info-type\"]“

```
<h2 class="sd-info-type">Roubo </h2>
```

"//*[@class=\"sd-info-title\"]“

```
<h1 class="sd-info-title">Fui assaltado por um motoqueiro</h1>
```

"//*[@class=\"obj-label valign-middle\"]"

```
<span class="obj-label valign-middle">Celular</span>
<span class="obj-label valign-middle">Documentos</span>
<span class="obj-label valign-middle">Cartão de Crédito</span>
<span class="obj-label valign-middle">Outros</span>
```

Caso de estudo: ‘ondefuiroubado.com.br’ -2-

"//*[@class=\"sd-info-data-hora\"]“

```
<span class="sd-info-data-hora"><span class="icon calendar"></span> 19/06/2013<span class="icon time"></span> 19:45</span>
```

"//*[@class=\"description valign-top\"]“

```
<h3 class="description valign-top">Estava me aproximando do portão de casa quando fui abordada por um motoqueiro de capacete, ele me segurou por trás, colocou a arma em minha cabeça e pediu que eu passasse a bolsa, até a chave de casa ele levou.</h3>
```

Caso de estudo: ‘ondefuiroubado.com.br’ -3-

```
occurrence_data <- data.frame(n_occurrence, latitude,
longitude, city, occurrence_type, occurrence_title, spoil,
date_time, occurrence_description)

dataset <- rbind(dataset, occurrence_data)
```

Caso de estudo: ‘ondefuiroubado.com.br’ - Melhores práticas -

- Para deixar o código mais robusto e possibilitar a recuperação das informações de forma mais fácil, algumas boas ideias na hora da construção do *script*
  - Guardar os dados em pedaços pequenos ao longo da obtenção (1000 por arquivo, por exemplo)
  - Permitir que se possa pegar um pedaço em específico (do 1001 até 2000, por exemplo)
  - Permitir que se possa recuperar ocorrências que não foram obtidas devido à um erro
  - Permitir atualizações incrementais aos dados armazenados
  - Guardar dados de log para conferir possíveis erros de execução e o horário de cada obtenção

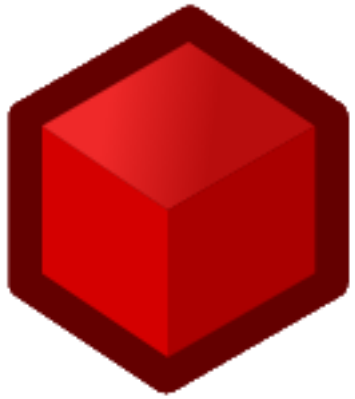
Caso de estudo: ‘ondefuiroubado.com.br’ - Melhores práticas -

- (continuação)
  - Organizar os dados e os logs em pastas próprias
  - Lidar com os possíveis objetos roubados de forma dinâmica
  - Lidar com possíveis valores faltando no campo de descrição
  - Definir um limiar de erro para interromper a execução (e impedir que rode para sempre)
  - Lidar com erros para permitir uma varredura contínua (sem interrupção à cada erro encontrado)
  - Nomear os arquivos com informações de data e hora em que foram obtidos

Referências

- http://en.wikipedia.org/wiki/Web_scraping
- http://en.wikipedia.org/wiki/Contact_scraping
- http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)
- http://www.w3schools.com/xpath/xpath_syntax.asp
- http://www.ondefuiroubado.com.br/

“Das leben ist zu kurz um deutsch zu lernen“ (Oscar Wilde)



CCD-UFF

Clube de Ciência de Dados