Web Scraping en R

SICSS - Montréal 2023

Alex Luscombe

14/06/2023

Table des matières

Comprendre le Web Scraping

Comprendre le Web Scraping

- Web Scraping : le processus automatisé d'extraction de données à partir de sites web
- En R, nous utilisons couramment les bibliothèques `rvest` et `RSelenium` pour accomplir cela
- Le web scraping pour la recherche en sciences sociales ouvre de nombreuses opportunités, mais il présente également de nombreux défis

Défis et opportunités du Web Scraping

- **Opportunités** :
  - Des données nouvelles et accessibles
  - Informations en temps réel
  - Recherche à méthodes mixtes
  - Analyse comparative
- **Défis** :
  - Considérations éthiques et légales
  - Complexité technique
  - Qualité des données et biais

Considérations légales et éthiques

- *Robots.txt* : Respectez les directives spécifiées dans le fichier robots.txt du site web pour les autorisations et les restrictions de scraping
- User-agents et en-têtes : Personnalisez les user-agents et les en-têtes pour fournir des informations d'identification pertinentes dans les requêtes de scraping
- Conditions d'utilisation : Examinez et respectez les conditions d'utilisation du site web qui est scrapé

Considérations légales et éthiques

- Restrictions d'utilisation des données : Comprenez et respectez les éventuelles limitations ou restrictions sur l'utilisation des données extraites
- Éthique : Tenez compte des préoccupations liées à la vie privée, de l'impact sur le site web ou la source de données, et respectez les bonnes pratiques de scraping éthique

/robots.txt

- *Robots.txt* : un fichier texte situé à la racine d'un site web qui fournit des instructions aux robots d'exploration web, y compris les bots de scraping, sur ce qui peut et ne peut pas être accédé
- Identifier le fichier robots.txt : Recherchez le fichier robots.txt en ajoutant << /robots.txt >> à l'URL du site web

Jargon du fichier /robots.txt

- `User-agent` : Spécifie le robot d'exploration ou user-agent auquel les règles s'appliquent
  - << * >> (wildcard) : S'applique à tous les robots d'exploration web
  - << User-agent: [user-agent spécifique] >> : Concerne un robot d'exploration web ou bot de scraping spécifique

Jargon du fichier /robots.txt

- `Disallow` : Liste les répertoires ou fichiers qui ne doivent pas être explorés ou scrapés
- `Allow` : Spécifie les exceptions aux règles de désactivation, autorisant des répertoires ou fichiers spécifiques à être explorés ou scrapés
- `Crawl-delay` : Respectez le délai spécifié pour éviter de surcharger le serveur avec trop de requêtes en un laps de temps court

Un exemple de fichier robots.txt

```
User-agent: Bingbot
Disallow: /

User-agent: *
Crawl-delay: 5
Disallow: /private/
Disallow: /admin/
Disallow: /confidential/
Allow: /public/
```

- Exemple concret : **https://www.utoronto.ca/robots.txt**
- Noter que tous les sites web ne disposent pas d'un fichier robots.txt : **https://sicss.io/robots.txt**

Bases de HTML

Bases de HTML

HTML : HyperText Markup Language

- Langage de balisage utilisé pour structurer et présenter le contenu sur le web
- Les balises définissent les éléments HTML, tels que les titres, les paragraphes, les listes, etc.
- Les éléments peuvent être imbriqués les uns à l'intérieur des autres pour créer une structure hiérarchique

```
<div class="container">
  <h1 id="titre-presentation"> Bienvenue dans ma présentation </h1>
  <p> Ceci est un exemple de texte en français </p>
</div>
```

Structure d'un document HTML

- `<html>` : Élément racine d'une page HTML
- `<head>` : Contient les métadonnées du document
- `<body>` : Représente le contenu de la page HTML

Inspection de la structure HTML

- Utilisez les outils de développeur du navigateur pour inspecter et analyser la structure HTML des pages web
- Accéder les outils de développeur : clic droit sur une page web, sélectionnez << Inspect >> pour les ouvrir
  - MacOS : `⌘ + Option + I`
  - PC : `Ctrl + Shift + I`

Sélection et extraction de données

Les sélecteurs CSS et XPath

- Les sélecteurs CSS permettent de cibler des éléments HTML spécifiques

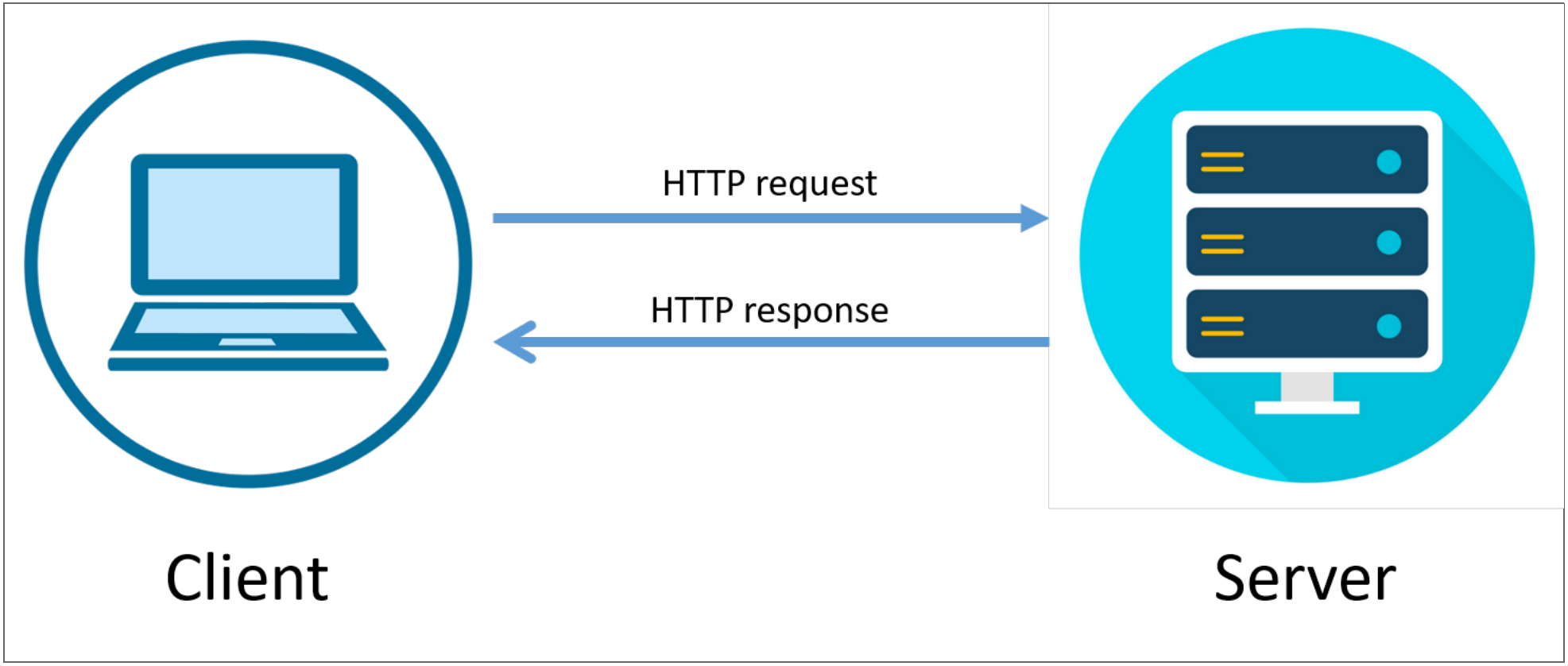
```html
<div class="container">
  <h1 id="titre-presentation"> Bienvenue dans ma présentation </h1>
  <p> Ceci est un exemple de texte en français </p>
</div>
```

- XPath est un langage d'expression pour naviguer dans les documents XML

```
//div[@class='container']/h1
```

- Outil utile : **SelectorGadget**

Extraction de données (GET request)



Source de l'image

Sélection et extraction avec rvest

```r
library(rvest)

# Spécifiez l'URL de la page web
url <- "https://www.example.com"

# Effectuez une requête pour obtenir le contenu HTML de la page web
# puis lisez-le - rvest::read_html le fait tout en un
webpage <- read_html(url)

# Extrait le contenu de l'élément HTML avec l'id="titre-presentation"
title_content <- webpage |>
  html_nodes("#titre-presentation") |>
  html_text()

# Extrait le contenu de l'élément HTML avec la classe="container"
container_content <- webpage |>
  html_nodes(".container") |>
  html_text()
```

Contenu dynamique

Contenu dynamique

- Les méthodes traditionnelles de scraping peuvent ne pas capturer efficacement le contenu dynamique
- Le contenu dynamique fait référence aux éléments des sites web qui changent ou se chargent dynamiquement à l'aide de `JavaScript`
- Les sites web dynamiques reposent sur le rendu côté client, ce qui rend l'extraction des données difficile
- Exemple : **https://sicss.io/people**

Contenu dynamique

- Deux solutions :
  - `API` : Explorez les APIs disponibles qui offrent un accès direct aux données dynamiques, nécessitant généralement clé d'API
    - Parfois, les APIs sont côté client, ce qui signifie qu'elles sont exécutées dans le navigateur du client plutôt que sur le serveur
  - `RSelenium`* : Paquet R avec des liens Selenium pour l'automatisation du web et le scraping de contenu dynamique (**Cela fonctionne mieux en Python* 🐍)

Le Web Scraping basé sur le Cloud (⚠️avancé!)

Le Web Scraping basé sur le Cloud

- Le Web Scraping basé sur le Cloud permet aux chercheurs d'automatiser et de collecter continuellement des informations
- Des exemples d'outils de Web Scraping basés sur le Cloud incluent GitHub Actions, AWS et Heroku
- Chacun de ces outils peut gérer l'exécution de scripts R côté serveur, permettant aux chercheurs d'utiliser ces plateformes pour exécuter des tâches de Web Scraping
- Tutoriel : **https://www.r-bloggers.com/2021/03/daily-stock-gainers-automated-web-scraping-in-r-with-github-actions/**

Questions et réponses

Pause de 5 minutes

Tutoriel