goo.gl/qik6dD

Web scraping et **APIs** avec **R**

à l'Usage des Sciences Sociales

François Briatte

Séminaire RUSS

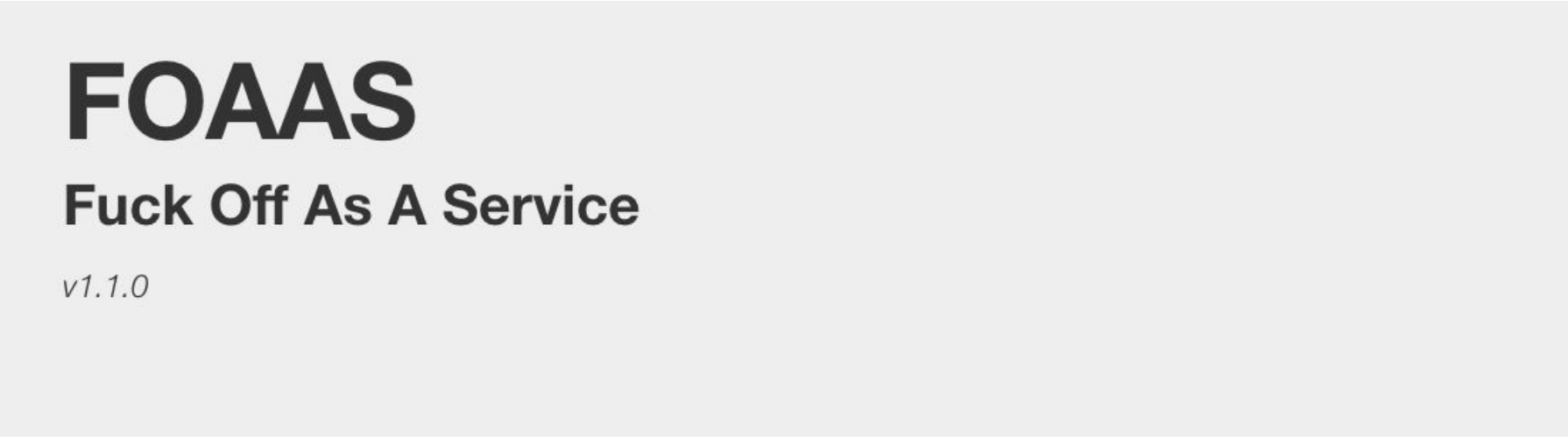
Mai 2017

Web scraping

Feature	Statistics	Machine Learning
Theoretical basis	Probability theory	"Hey, I wonder if this will work??"
Measurement	Should be careful and consistent	Whatever
Foundational result	Central limit theorem	Web scraping
Distribution naming conventions	Long dead European males	Distributions?

Source: Phil Schrodt

API (Application programming interface)



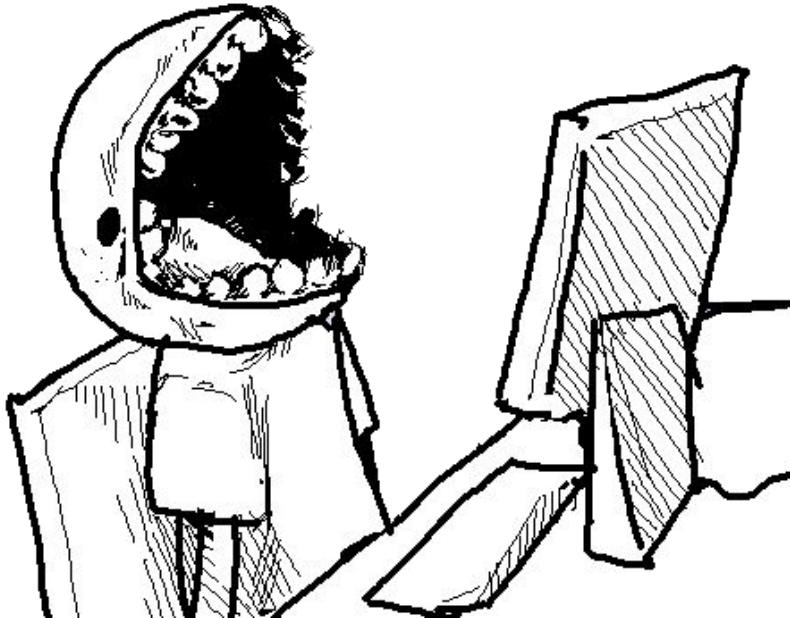
Hors d'oeuvre

Récupération de données d'enquêtes

Deux projets d'Anthony Damico à connaître :

- Analyze Survey Data for Free (avec R + MonetDBLite)
- lodown · *locally download and prepare publicly-available microdata*

Un des meilleurs exemples d'utilisation de R pour accéder à des données d'enquêtes.
Tout est déjà codé :-)



Menu

Merci de poser vos questions
au fil de la présentation

La deuxième partie requiert **R**
+ une connexion Internet

Contexte

Stratégies

Outils

Syntaxes

25 *slides*

Interfaces

Scraping

Références

25 *slides*
∈ *exemples*

Contexte

- **Augmentation de la disponibilité des données** dans des formats facilement téléchargeables et manipulables
- **Volontés institutionnelles de faciliter la réutilisation** de *certaines* données : 

/ gov. / science – ex. OPEN (Open, Public, Electronic and Necessary) Government Data Act, USA
- **Avancées dans les formats, licences et standards** d'exploitation de fichiers – ex. CSVY, NDJSON, CC, ODbL
- **Montée en capacité des infrastructures et outils Web** – (connexions, serveurs, interfaces interactives, etc.)

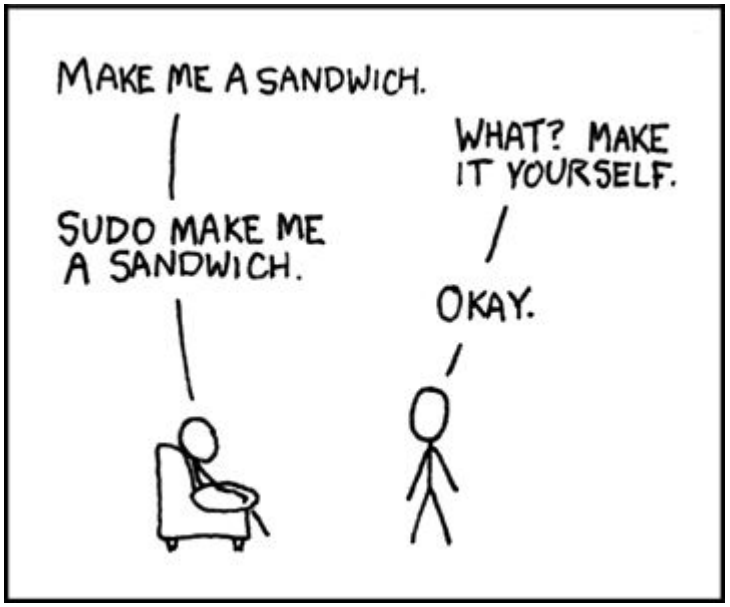
Contexte

Approximately 10% of [the 206 new packages that arrived on CRAN in January] have to do with **providing access to data** in by some means or another. Some packages contain the data sets, some provide **wrappers to APIs**, and at least one package provides **code to scrape** data from a site.

Joseph Rickert, January 2017

Contexte

- **Faible compétence collective** en matière de collecte de données sur Internet
- **Avantage comparatif immédiat** très rapidement rentabilisable dans les milieux scientifiques
- **Problème connexe** : dépendance aux personnels compétents
- **Problème principal** : forte volatilité des données numériques



source · trope

Contexte

Scraping : applications **scientifiques**, applications **militantes**

- **Secteur public et corruption :** en Roumanie, l'Agence Nationale de l'Intégrité publie « 7 millions » de déclarations d'intérêts – mais les publie-t-elle vraiment toutes ?
- **Secteur privé et illégalité :** aux États-Unis, AirBnb publie ses bases utilisateurs – mais sont-elles complètes ?

> **Murray Cox** and **Tom Slee**, who had independently been scraping **Airbnb** data before the company publicly released what they said was all of their data, has evidence that **the home-sharing company tried to cover up potentially illegal postings**. Be wary of (data) sharing that seems too good to be true.

NYU Data Science Community Newsletter #90, Feb 17, 2017

Contexte

UCLA iSchool faculty **scraped** **climate science data** from federal websites, fearing it would disappear under Trump's administration. A similar event is taking place at **NYU** on February 4th and both were inspired by a hackathon at the **University of Toronto** to preserve US environmental data last December.

NYU Data Science Community Newsletter #87, Jan 27, 2017

Los Alamos National Lab **released** 16 years of GPS solar weather data thanks to the Obama administration. Let's agree to scrape this from data.gov (search "GPS energetic particles" and store it locally.

NYU Data Science Community Newsletter #88, Feb 3, 2017

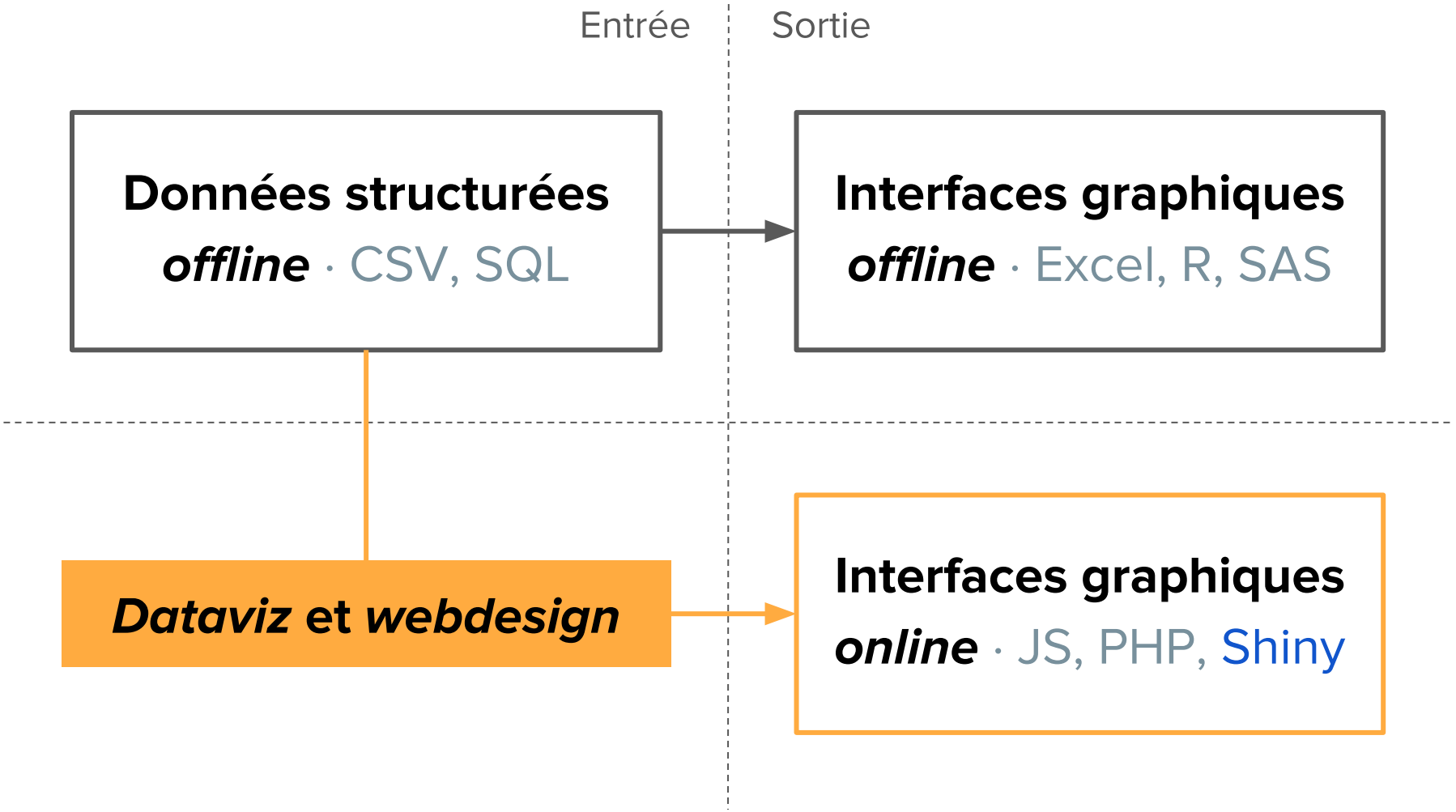
John Rozsa, a graduate student at **Eastern Michigan University**, built **EPA Data Dump** to **house** all of the EPA data he scraped from the federal website to ensure researchers will have access, even if the agency is terminated. Thank you, John. Let us know if you need help.

NYU Data Science Community Newsletter #91, Feb 24, 2017

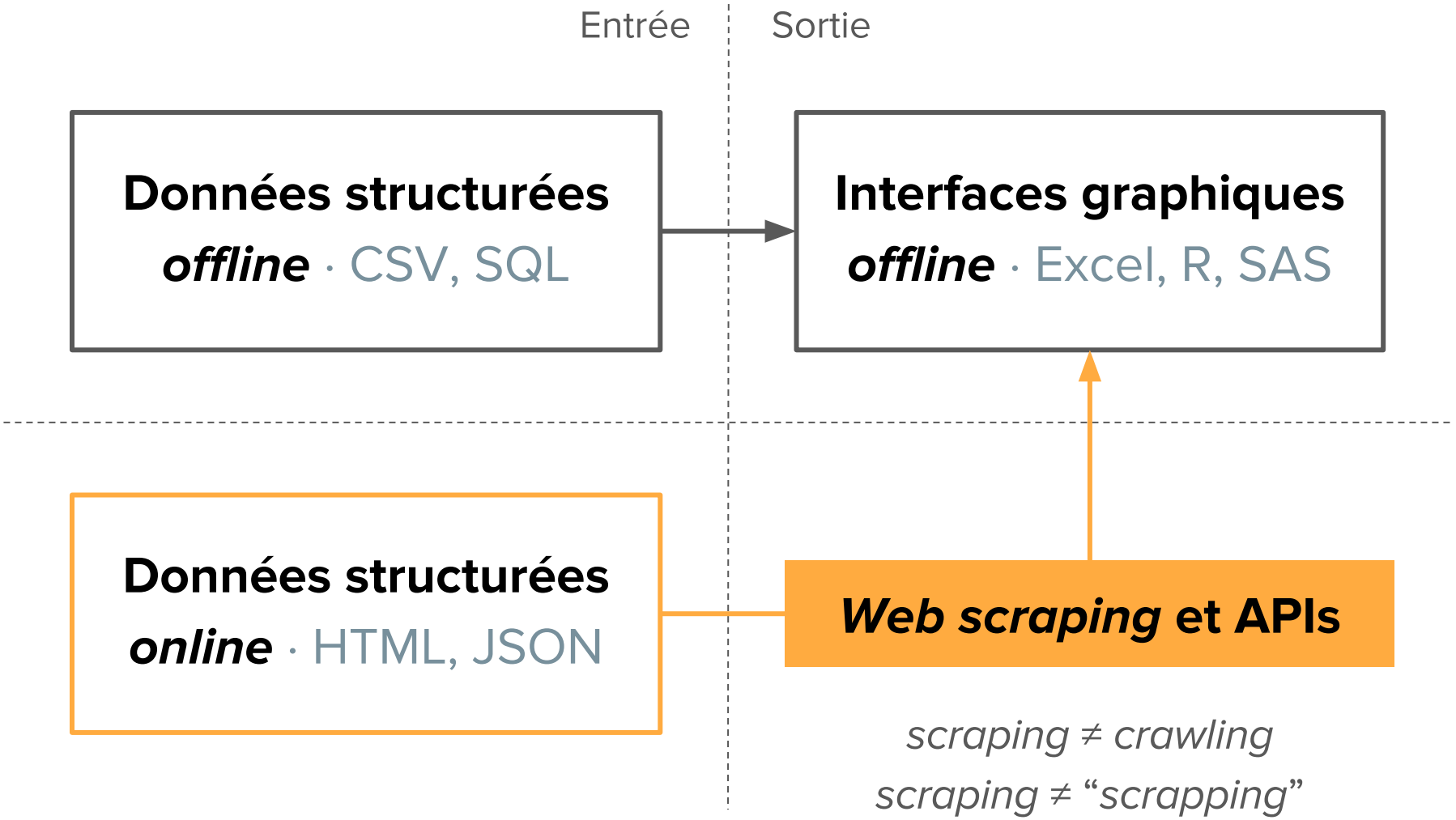
Stratégies

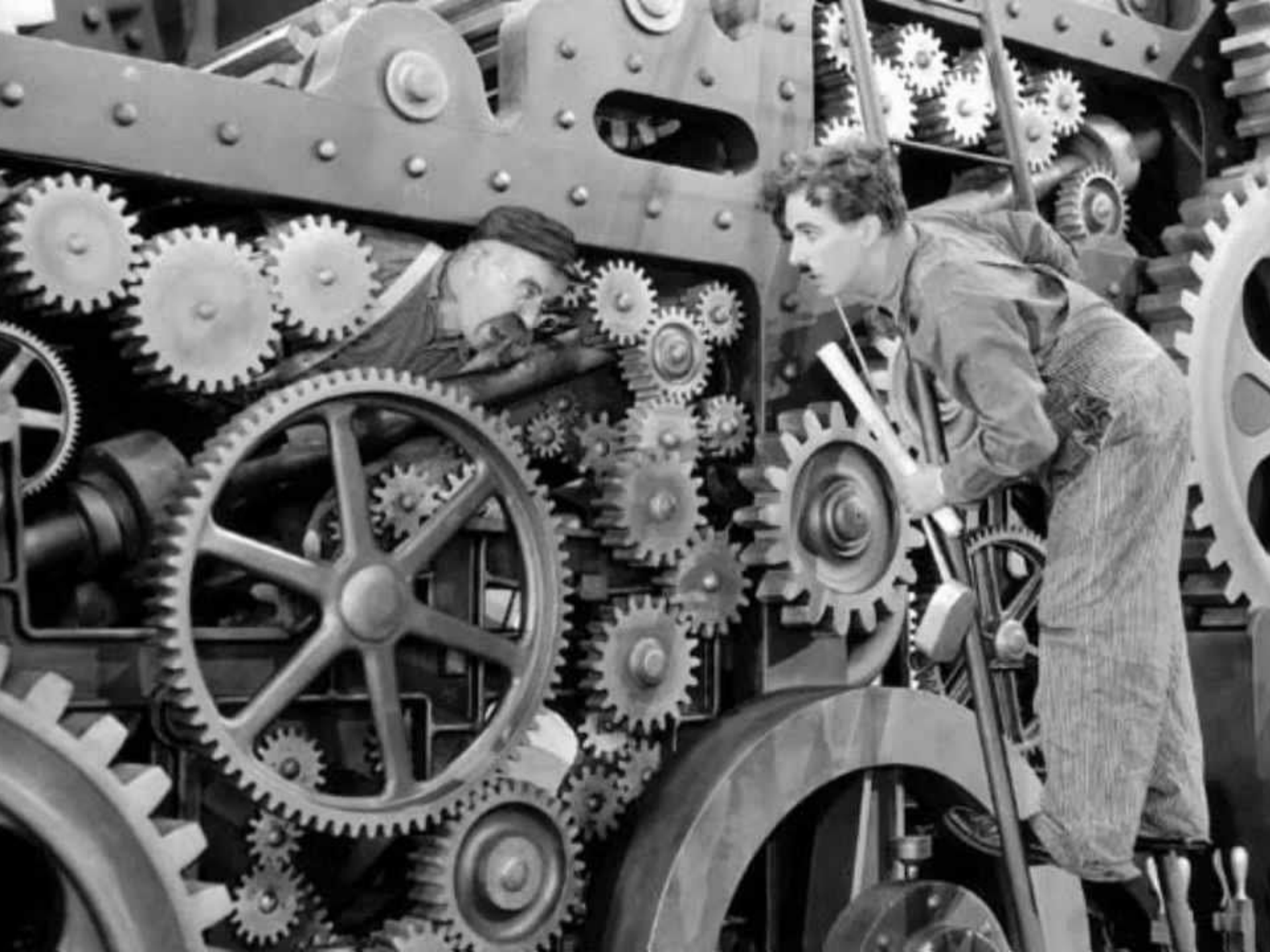
	Entrée	Sortie
	Données structurées	**Interfaces graphiques**
	offline · CSV, SQL	***offline*** · Excel, R, SAS
	Données structurées	**Interfaces graphiques**
	online · HTML, JSON	***online*** · JS, PHP, Shiny

Stratégies


Entrée
Sortie
Données structurées
offline · CSV, SQL
Interfaces graphiques
offline · Excel, R, SAS
Dataviz et *webdesign*
Interfaces graphiques
online · JS, PHP, Shiny

Stratégies

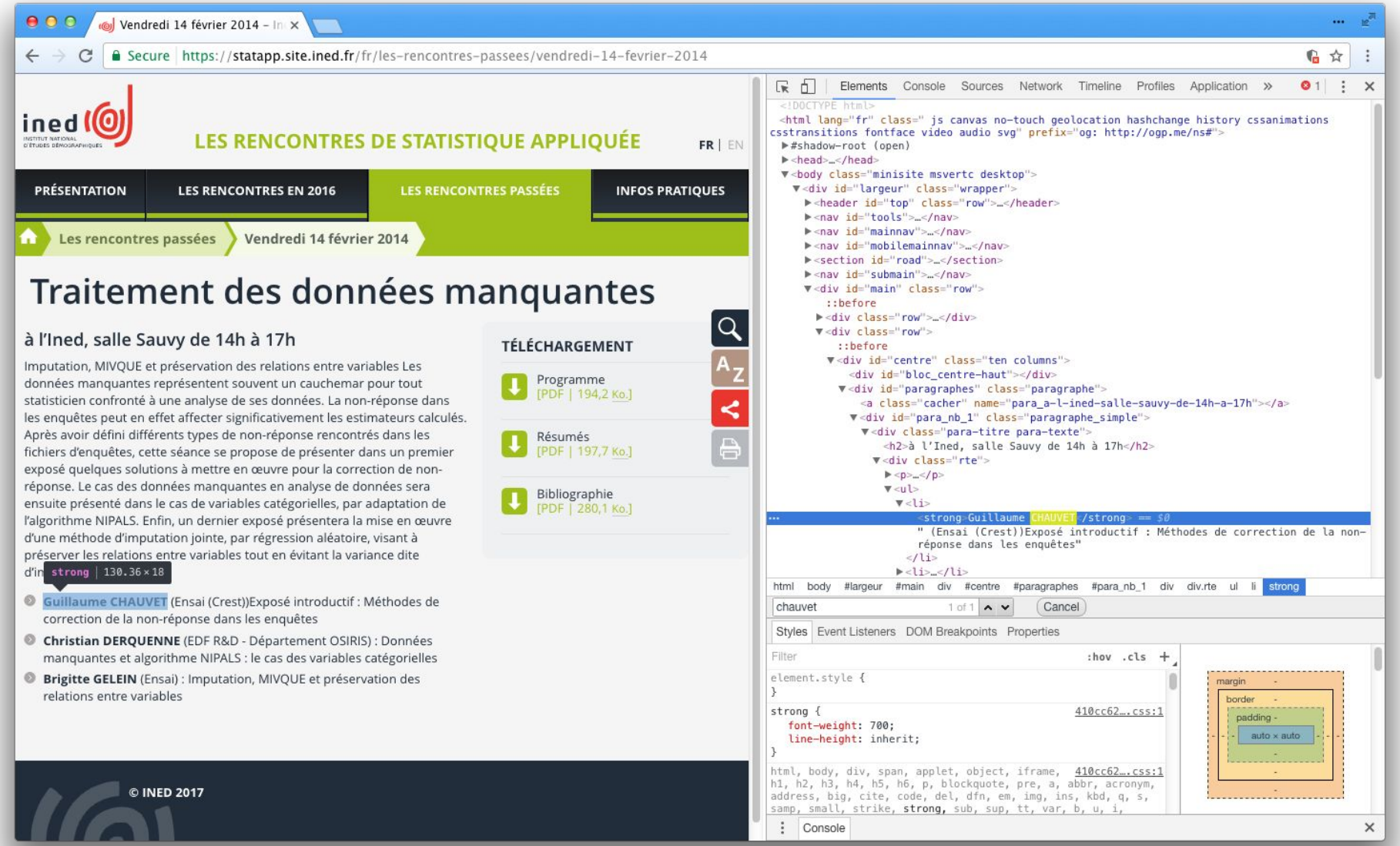

Entrée
Sortie
Données structurées
offline · CSV, SQL
Interfaces graphiques
offline · Excel, R, SAS
Données structurées
online · HTML, JSON
Web scraping et APIs
scraping ≠ crawling
scraping ≠ “scrapping”



Outils

- **Navigateurs Web : Developer Tools** – source HTML et éléments, accès au DOM, console JavaScript
- **Utilitaires : curl** + **wget** – téléchargement, tests de connexion sous différents protocoles ou user agents
- **Syntaxes : HTML** + **CSS**, XPath (XML), CSSSelect, expressions régulières, formats de fichiers, langages *server-side*, langages de requête (*cf.* section suivante)
- **Packages R : httr**, **rvest** + **xml2**, stringr, lecteurs de formats (ex. jsonlite, readr), *wrappers* d'APIs

Developer Tools (Chrome, Firefox, etc.)



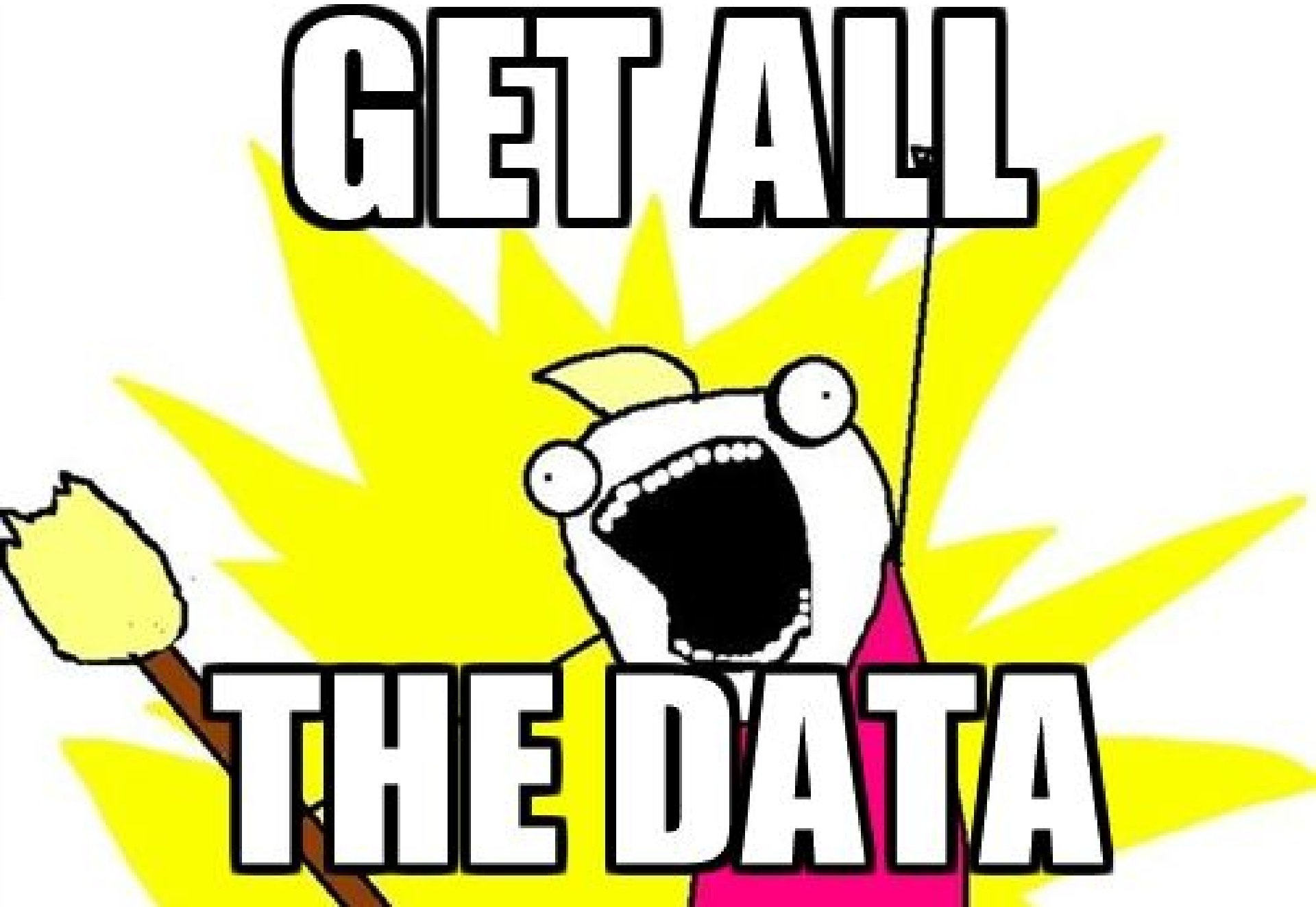
Syntaxes HTML / CSS / XPath / CSSSelect

```
143     <div id="main" class="row">
144         <div class="row">
145             <div id="bloc_avant-main" ><div class="bloc " id="bloc_avant-main_titre"><div id
    class="clearfix"><h1 id="h1" class="misob ">Traitement des données manquantes</h1></div></di
    </div>
146         <div class=
147
148                 <di
    class="cacher" name
    class="paragraphe_s
149     <h2>à l'Ined, s
150             <p>Impu
    représentent souver
    réponse dans les er
    avoir défini différ
    propose de présente
    non-réponse. Le cas
    variables catégorie
    mise en œuvre d'une
    relations entre variables tout en évitant la variance dite d'imputation.</p>
151 <ul>
152 <li><strong>Guillaume CHAUVET</strong> (Ensai (Crest))Exposé introductif : Méthodes de corre
    non-réponse dans les enquêtes</li>
153 <li><strong>Christian DERQUENNE</strong> (EDF R&D - Département OSIRIS) : Données manquantes
    algorithme NIPALS : le cas des variables catégorielles</li>
154 <li><strong>Brigitte GELEIN</strong> (Ensai) : Imputation, MIVQUE et préservation des relati
    variables</li>
155 </ul>         </div>
156         </div>
```

```
1  library(rvest)
2  library(stringr)
3
4  b <- "https://statapp.site.ined.fr/fr/"
5  p <- str_c(b, "les-rencontres-passees/vendredi-14-fevrier-2014")
6  read_html(p) %>%
7    html_nodes("#main strong") %>%
8    html_text

[1] "Guillaume CHAUVET"   "Christian DERQUENNE" "Brigitte GELEIN"
```

Wget pour récupérer un cours complet

```
wget \
    --directory-prefix=/Users/fr/Desktop \
    -c \
    --proxy=off \
    -Q0 \
    --passive-ftp \
    -r \
    -l6 \
    --no-parent \
    http://www.stat.cmu.edu/~cshalizi/uADA/16/
```


GET ALL
THE DATA

Outils

Autres aspects à considérer

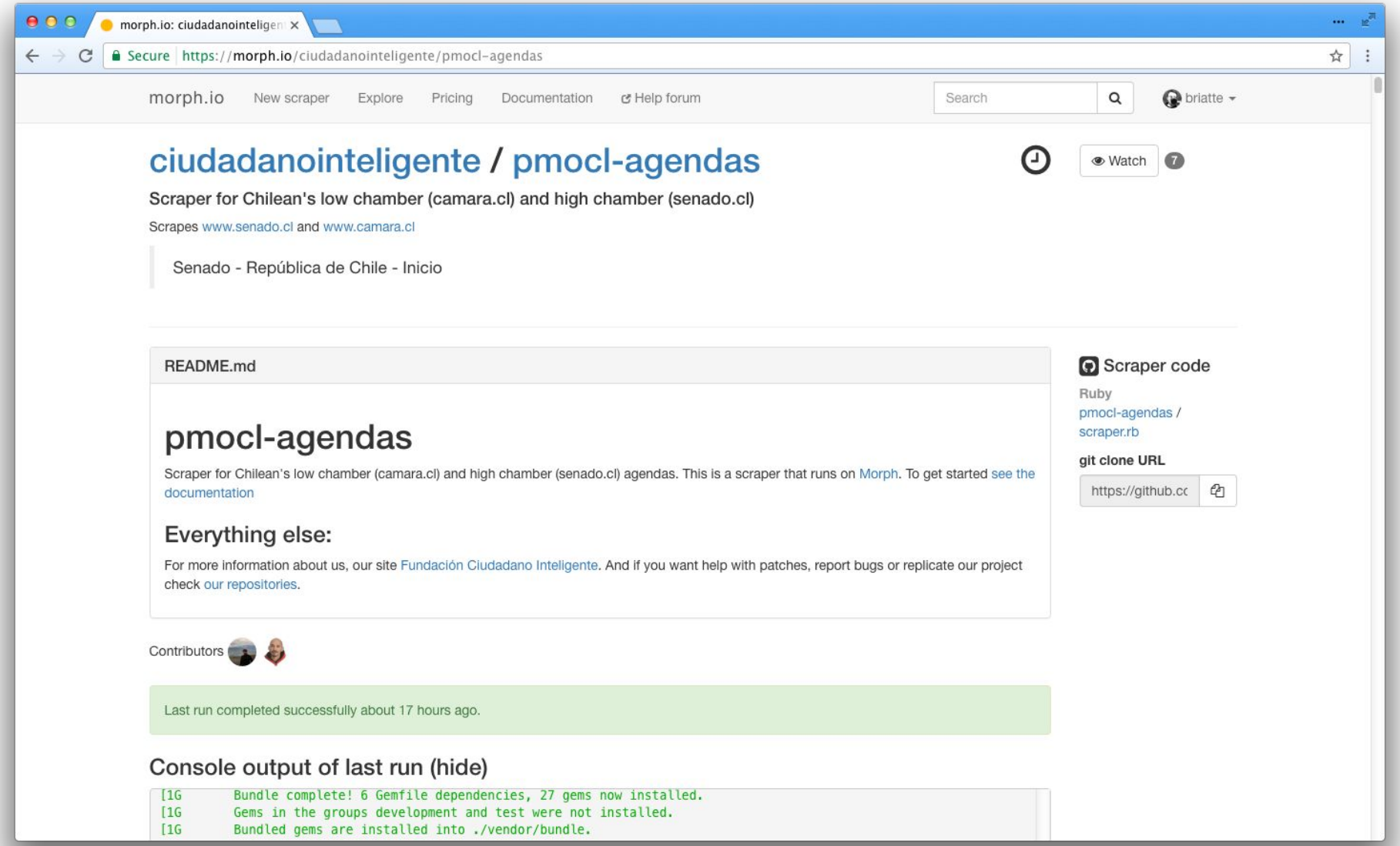
Alternatives

- JavaScript (CasperJS, PhantomJS, SlimerJS, headless Chrome)
- Python (BeautifulSoup, lxml, Scrapy, Facebook Page Post Scraper)
- Ruby (Nokogiri), Perl (WWW::Mechanize), PHP, ...

Automation

- Cron jobs
- Morph.io · ex-ScraperWiki

Morph.io (Python, Ruby, PHP, Perl, Node.js)



Outils

Autres aspects à considérer

Hardware

- Connexion Internet robuste + VPN (parfois utile…)
- Mémoires (cache, disque) rapides + laisser du *lag* serveur

Légalité

- Conditions d’utilisation des sites Internet
- Conditions de redistribution des données
- Données personnelles (ex. adresses email, photos)

WHEN PROGRAMMERS SCRAPE BY

By Rusty Foster February 10, 2014

"Scraping" is not a word with a lot of positive connotations. In "A Christmas Carol," Charles Dickens describes Ebenezer Scrooge as "a squeezing, wrenching, grasping, scraping, clutching, covetous old sinner." We feel the impropriety of Scrooge's scraping all the more keenly for his wealth and power, since scraping is most often an act of the weak or desperate; we talk of "scraping by" or "scraping the bottom of the barrel," and of "bowing and scraping" to someone more powerful. It's also recently become news: Aaron Swartz, Chelsea Manning, and Edward Snowden have all found themselves in trouble during the past few years for something called "scraping."

Unnamed intelligence officials, describing how Edward Snowden collected his archive of N.S.A. files, told the *Times*, "We do not believe this was an individual sitting at a machine and downloading this much material in sequence" and that the process was "quite automated." The officials didn't specify what software Snowden used, but "said it functioned like Googlebot, a widely used web crawler that Google developed to find and index new pages on the web." The difference between a "crawler" and a "scraper" is subtle, but typically a crawler is smarter about the links it follows, what it downloads, and what it leaves uncopied. For the most part, though, "crawling" is just scraping with a fancier name, and Google created one of the world's most valuable companies in part by being better at scraping than anyone else. Google was incorporated in 1998, and by 2002 its Web-scraping "Googlebots" were so ubiquitous and voracious that, in a short story titled "Robot Exclusion Protocol," the programmer and writer Paul Ford imagined one trying to index his bathroom. Some have suggested that Google's recent acquisition of the smart-device maker Nest Labs is effectively an effort to scrape real-world data about our homes and lives, to add to the company's trove of information about us, which now includes information about the Web pages we visit, our e-mails, the books we read, our shopping habits, and more.

Someone scraped 40,000 Tinder selfies to make a facial dataset for AI experiments

40 000 photos de profil prélevées sur Tinder publiées en ligne

Un utilisateur a automatiquement « aspiré » ces photos afin d'en faire une base de données servant à développer des programmes d'intelligence artificielle.

Syntaxes

- **HTML** + **CSS** **:** référentiels W3C et DOM
- **XPath** **:** exploration d'arborescences XML
- **CSSSelect** **:** conversion de sélecteurs CSS vers XPath

Et aussi

- **Expressions régulières :** sélection de chaînes de texte
- **Formats de fichiers** (ex. JSON, RDF, XML)
- **Langages *server-side* :** générateurs DHTML (ex. ASP, PHP)
- **Langages de requêtes** (ex. SQL, SPARQL)


Questions ?

Scraping : récupérer les données

Approche manuelle

- **Lecture des sources HTML** (avec Developer Tools)
- **Requêtes HTTP GET** (via rvest ou via httr si cookies)
- **Boucles de téléchargement** (avec gestion des erreurs)

Autres approches

- *Scrapers* interactifs (ex. Scraper pour Google Chrome)
- *Headless browsers* (ex. PhantomJS, Selenium, Chrome)

Spécification W3C (draft) : WebDriver

Packages R : seleniumPipes, RSelenium, webdriver (draft)

(en cours)

Scraping : reconstituer les données

Lecture

- ***Parsing*** – réencodage (UTF-8), extraction des champs
- **Nettoyage** – conversions, corrections, standardisations

Stockage

- **CSV** (fichiers intermédiaires, interopérables)
- **RDS** (objets finalisés, compressés)
- **Base de données**

Drivers monetdb : MonetDB.R, MonetDBLite (présentation)

Drivers SQL : RMySQL, RPostgreSQL, RSQLite · interfaçables avec dplyr

Scraping : exemple pour séminaRistes

Récupérer les noms des intervenants des séminaires Ined ‘RUSS’ et ‘STATAPP’ (rencontres de statistique appliquée)

- **Code R : Gist** (200 lignes, quelques minutes d’exécution)
- **Dépendances :** dplyr, genderizeR, readr, rvest, stringr

Protocole

- Récupérer la liste des séances (*seminars*)
- Récupérer la liste des intervenants (*speakers*)
- Imputer le sexe des intervenants et estimer la proportion d’intervenants féminins

Scraping : exemple pour vulcanologues

```
# Récupération d'éruptions volcaniques
# Source : Bob Rudis, http://stackoverflow.com/a/42100715

library(httr)  # GET, POST
library(rvest) # html_*
library(purrr) # map
library(dplyr) # select

# émulation du formulaire de la page...
x <- POST("http://volcano.si.edu/search_eruption_results.cfm",
          body = list(bp = "", `eruption_category[]` = "",
                      `country[]` = "", polygon = "",  cp = "1"),
          encode = "form")
```

Scraping : exemple pour vulcanologues

```
# Récupération d'éruptions volcaniques
# Source : Bob Rudis, http://stackoverflow.com/a/42100715

# ... et extraction de la table de résultats :
content(x, as = "parsed") %>%
  html_nodes("div.DivTableSearch") %>%
  html_nodes("div.tr") %>%
  map(html_children) %>%
  map(html_text) %>%
  map(as.list) %>%
  map_df(setNames, c("volcano_name", "subregion", "eruption_type",
                     "start_date", "max_vei", "X1")) %>%
  select(-X1)
```

Interfaces

Définition

- **API** : *Application Programming Interface* (cf. Awesome API)
- **Entrée** (ex. URL, SPARQL) ➜ **Sortie** (ex. CSV, JSON, XML)
- Avec ou sans **authentification** (ex. OAuth · cf. OAuth Bible)

Exemples

- Banque Mondiale (ouverte ; sorties JSON et XML)
- Regards Citoyens (ouverte ; sorties CSV, JSON et XML)
- Twitter (via OAuth ; REST et Streaming ; sorties JSON)

Interfaces

Catalogues

- News API
- Public APIs

voir aussi les packages R de journaux
ex. GuardianR / rdian, nytimes

exemple bizarre : nazihuntR – *lolwat*

Packages R

- Web Technologies and Services (CRAN Task View)

  Ex. oecdr, WDI (World Development Indicators)

  Ex. Rfacebook, Rlinkedin, twitteR / RTwitterAPI / streamR / rtweet

  Ex. rwikidata, WikidataQueryServiceR, WikiDataR, WikipediR

- SPARQL (pour communiquer avec un *endpoint* SPARQL)

Exemple de conditions d’utilisation

1. Limitation of Liability for API Use

2. No Abuse or Overuse of the API

Abuse or excessively frequent requests to GitHub via the API may result in the temporary or permanent suspension of your account's access to the API. GitHub, in our sole discretion, will determine abuse or excessive usage of the API. We will make a reasonable attempt to warn you via email prior to suspension.

It is prohibited to use the API to download data or Content from GitHub for spamming purposes, including for the purposes of selling GitHub users' personal information, such as to recruiters, headhunters, and job boards.

All use of the GitHub API is subject to these Terms of Service and the GitHub Privacy Statement.

GitHub may offer subscription-based access to our API for those Users who require high-throughput access or access that would result in resale of GitHub's Service.

3. GitHub May Terminate Your Use of the API


RTFM

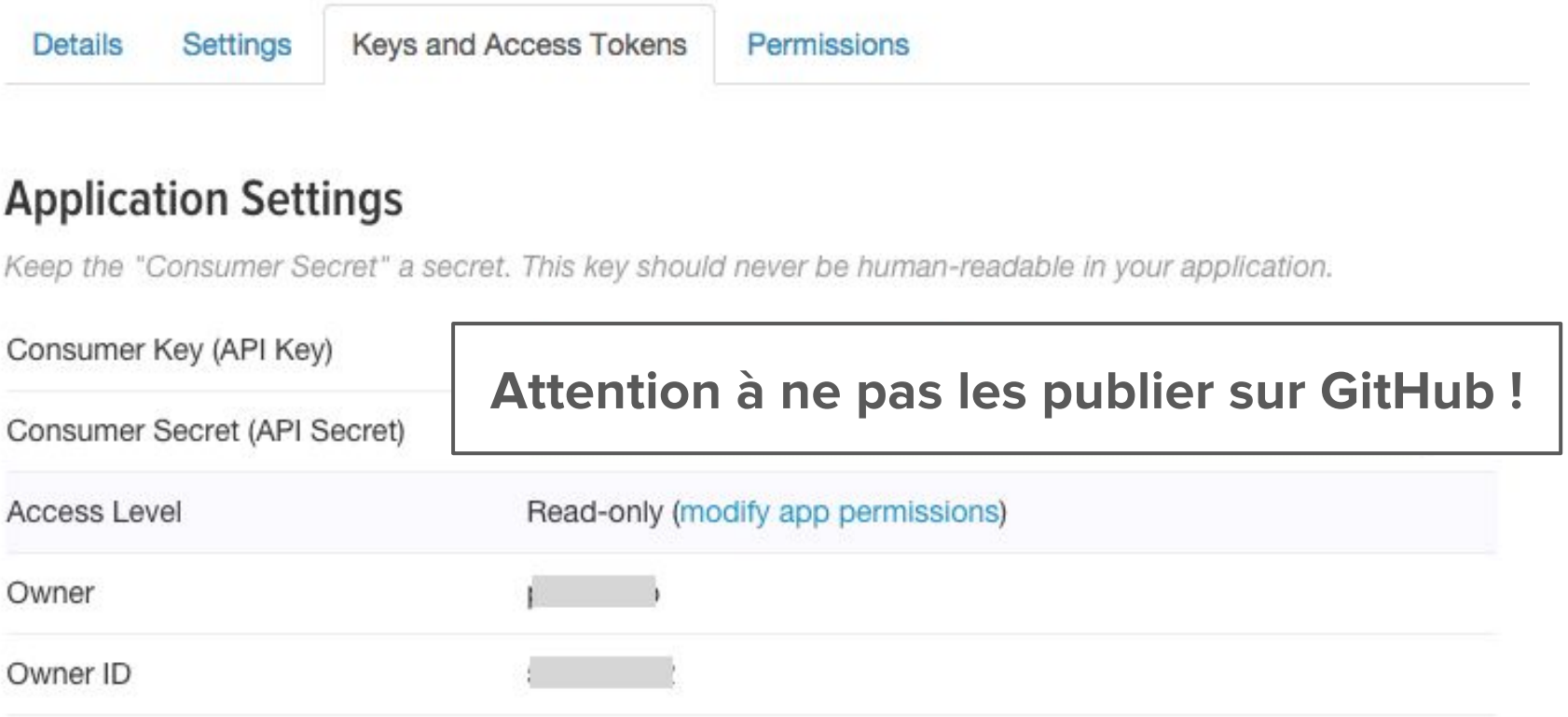
Interfaces : exemple GET / R

```r
# Imputation du sexe à partir du prénom
# API : genderize.io

# Avec genderizeR
library(genderizeR) # imports magrittr
findGivenNames("benoit")

# Avec httr
library(httr)
GET("https://api.genderize.io/?name=benoit") %>%
    content() # formatted from JSON
```

Interfaces : exemple OAuth / R

Récupération de tweets à partir de mots-clés · détails

Étape 1 : créer une application sur Twitter Apps



Interfaces : exemple OAuth / R

Étape 2 : créer un objet d'authentification

```
library(ROAuth)

requestURL     <- "https://api.twitter.com/oauth/request_token"
accessURL      <- "https://api.twitter.com/oauth/access_token"
authURL        <- "https://api.twitter.com/oauth/authorize"
consumerKey    <- "#########" # add your consumer key here!
consumerSecret <- "#########" # add your consumer secret here!

oauth <- OAuthFactory$new(consumerKey    = consumerKey,
                          consumerSecret = consumerSecret,
                          requestURL     = requestURL,
                          accessURL      = accessURL,
                          authURL        = authURL)
```

Interfaces : exemple OAuth / R

Étape 3 : lancer le processus d'authentification

```
# locate CAcert.org trusted certificates
cacert <- system.file("CurlSSL", "cacert.pem", package = "RCurl")

# executing this line will open a browser window
oauth$handshake(cainfo = cacert)
```


```
# save the 'certified' OAuth object
saveRDS(oauth, file = "oauth.rds")
```

Interfaces : exemple OAuth / R

Étape 4 : récupérer le Public Stream en morceaux (*chunks*)

```
library(streamR)

a <- readRDS("oauth.rds") # credentials
q <- c("#hashtags", "and", "other", "#search", "keywords")

for (i in (15 * 24 * 2):1) { # run for 15 days

  f <- paste0("tweets", sprintf("%03.0f", i), ".json")

  # capture matching public tweets for 25 minutes
  try(filterStream(f, track = q, timeout = 60 * 25, oauth = a))

  # sleep for 5 minutes to avoid choking the API
  Sys.sleep(60 * 5)

}
```

Interfaces *vs.* **Scraping**

Pour récupérer des données Wikipédia :

Approche faiblement structurée – récupération du code HTML (souvent irrégulier) des pages-source :

Ex. Extracting notable deaths from Wikipedia

Ex. The U.S. has been at war 222 out of 239 years

Inconvénient Pr (*code breaks*) ≫ 0

Interfaces *vs.* **Scraping**

Pour récupérer des données Wikipédia :

Approche fortement structurée – utiliser Wikidata et son *endpoint* SPARQL pour récupérer toutes les entités :

Ex. Importation de données Wikidata dans Gephi

Ex. Tableaux de Vermeer montrant des cartes (source)

Inconvénient Syntaxe SPARQL

Interfaces + **Scraping**

Certains types de données requièrent de pouvoir **combiner *scraping* et appels d'API** :

Ex. Analyse de sentiment sur **données musicales**

- Albums Last.fm · REST ou XML-RPC, i.e. HTTP + XML
- Albums Spotify · OAuth2
- Paroles Genius · OAuth2

Applications (avec R) : Radiohead, Kanye West

Interfaces + *Scraping*

```
# Extraction du texte principal d'une page HTML

# API : mercury.postlight.com/web-parser


library(httr)    # GET
library(rvest)   # read_html, html_text
library(stringr) # str_*

j <- GET(str_c("https://mercury.postlight.com/parser?url=",
               "https://www.monde-diplomatique.fr/",
               "2015/12/GOMBIN/54357")),
         add_headers("x-api-key" = "########")) # API key here!

content(j)$content %>%
  read_html %>%
  html_text %>%
  str_replace_all("\\s", " ") %>%
  writeLines("body.txt")
```

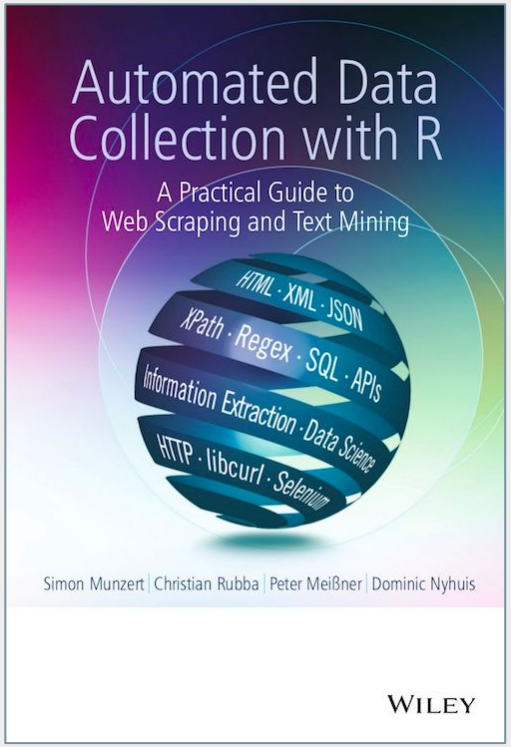
Références

- **Guides introductifs et tutoriels**

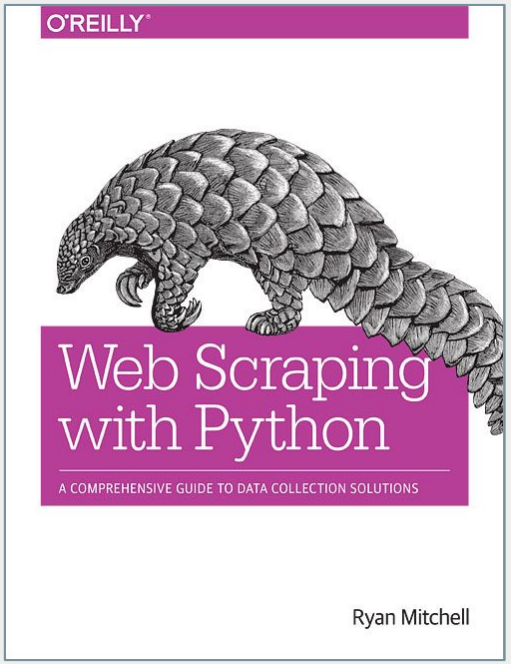
  [A Short Introduction to HTML
  Scraping for Craft Beers,
  Utilisation d'APIs natives…]

- **Cas pratiques**
- **Manuels**
- **Référentiels**
- **StackOverflow**

  [css, httr, phantomjs,
  regex, rvest, xpath…]



… et aussi



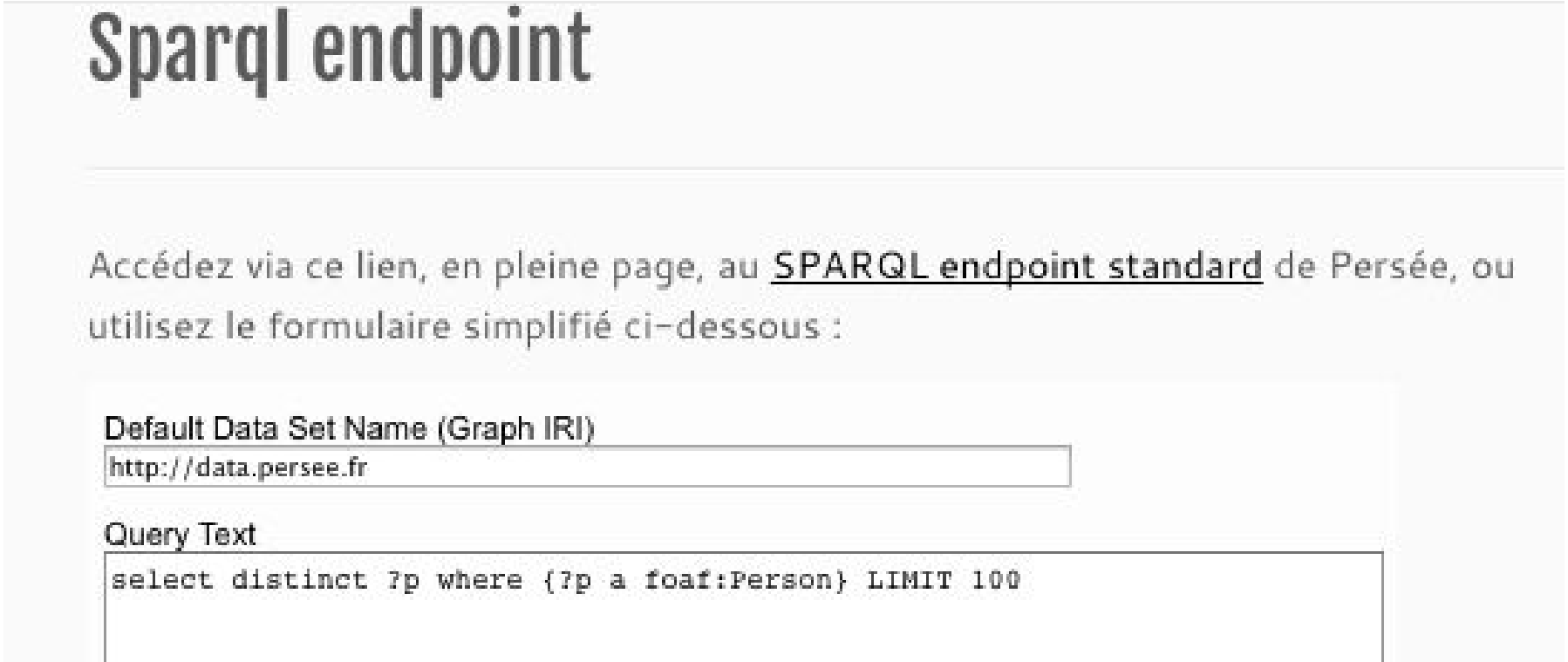
Lectures supplémentaires

- **Closing APIs and the public scrutiny of very large online platforms** – Bernhard Rieder
- **Global Open Data Index** – Open Knowledge Foundation
- **On Missing Data Sets** – Mimi Onuoha
- **rOpenSci** – “Transforming science through open data”
  - CRAN Task View: **Open Data**
  - CRAN Task View: **Web Technologies and Services**
- **Scraping Web sources: Two illustrations**

À vos heures perdues…

Utilisez les fichiers RDF/XML et/ou l'*endpoint* SPARQL de Persée pour réutiliser leurs données bibliographiques :

data.persee.fr/sparql (docs, schemas)



À vos heures perdues…

Demandez à Baptiste Coulmont s'il a besoin d'un coup de main pour récupérer l'une de ses sources de prénoms :

memoiredeshommes.sga.defense.gouv.fr

À vos heures perdues...

Aidez à récolter les investitures aux élections législatives (code en bash + Python) : données, script PS, scripts FI et EM

(avec bash : autoupdate ; avec Python : Morph.io)

```
#!/bin/bash

cd $(dirname $0)/..

bin/download.sh

if git status | grep "data/" > /dev/null; then
  if ! grep "[0-9]," data/investitures-deputes-PS-2017.csv > /dev/null ; then
    echo "WARNING: no result from http://www.parti-socialiste.fr/liste-candidats-aux-legislatives-
    exit 1
  fi
  git commit data -m "autoupdate"
  git push
fi
```

À vos heures perdues…

Apprenez à utiliser Mashape pour utiliser tout un tas d'APIs (comme ce détecteur de bots sur Twitter)

market.mashape.com/OSoMe/botometer