

WEEK 2: WEB DATA COLLECTION 2

SECU0050

BENNETT KLEINBERG

23 JAN 2020



Data Science for Crime Scientists

WEEK 2: WEB DATA COLLECTION 2

TODAY

- more HTML
- basic browser simulation

THREE ELEMENTS OF A WEBPAGE

1. Structure
2. Behaviour
3. Style

RAW HTML ARCHITECTURE

```
<!DOCTYPE html>
<html>
<body>

HERE COMES THE VISIBLE PART !!

</body>
</html>
```

TABLES

```
<table>
  <tr>
    <th>Departments</th>
    <th>Location</th>
  </tr>
  <tr>
    <td>Dept. of Security and Crime Science</td>
    <td>Division of Psychology and Language Sciences</td>
  </tr>
  <tr>
    <td>35 Tavistock Square</td>
    <td>26 Bedford Way</td>
  </tr>
</table>
```

HTML <TABLE>...</TABLE>

A screenshot of a web browser window displaying an HTML document. The browser interface includes standard navigation buttons (back, forward, search, home) and a URL bar showing the local file path: file:///Users/bennettkleinberg/GitHub/ucl_aca_20182019/slides/html_example.html.

The content of the page is as follows:

- I'm a heading at level 1**
- I'm the first paragraph
- I'm the second paragraph
- I'm the third paragraph
- [Click here to go to UCL's website](#)
- Departments**
Dept. of Security and Crime Science
35 Tavistock Square
- Location**
Division of Psychology and Language Sciences
26 Bedford Way

LISTS

```
<ul>
  <li>Terrorism</li>
  <li>Cyber Crime</li>
  <li>Data Science</li>
</ul>
```

I'm a heading at level 1

I'm the first paragraph

I'm the second paragraph

I'm the third paragraph

[Click here to go to UCL's website](#)

Departments

Dept. of Security and Crime Science
Division of Psychology and Language Sciences
35 Tavistock Square

Location

26 Bedford Way

- Terrorism
- Cyber Crime
- Data Science

IDENTIFYING ELEMENTS: IDs

Elements (can) have IDs:

```
<p id='paragraph1'>This is a paragraph</p>

```

Same for tables, links, etc.

Every element can have an ID.

You need unique IDs! Two elements cannot have the same ID.

IDENTIFYING ELEMENTS: CLASSES

Common elements (can) have CLASSES:

```
<p id="paragraph1" class="paragraph_class">I am the first paragraph</p>
<p class="paragraph_class">I am the second paragraph</p>
<p class="paragraph_class">I am the third paragraph</p>
```

Multiple elements can have the same class.

WEBSCRAPING IN PRACTICE

- FBI's missing persons
- Dodgy exotic animals trading page (tutorial)

FBI'S MISSING PERSONS

<https://www.fbi.gov/wanted/kidnap>

How could we explore the target page?

AIMS

1. Getting a list of all names
2. Storing the bio information
3. Downloading the poster

GETTING STARTED

Set up your workspace first:

```
library(rvest)
```

```
## Loading required package: xml2
```

```
target_url = 'https://www.fbi.gov/wanted/kidnap'
```

1. GETTING A LIST OF ALL NAMES

Access the full html page (snapshot-mode):

```
target_page = read_html(target_url)
target_page
```

```
## {xml_document}
## <html lang="en" data-gridsystem="bs3">
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
## [2] <body id="visual-portal-wrapper" class="  portaltypes-folder site-1">
```

1. GET A LIST OF ALL NAMES

Key here: look for the <h3> heading with class title:

```
all_titles = target_page %>%
  html_nodes('h3.title')

#note: equivalent to "html_nodes(target_page, 'h3.title')"
```

```
head(summary(all_titles))
```

```
##      Length Class     Mode
## [1,]    2   xml_node  list
## [2,]    2   xml_node  list
## [3,]    2   xml_node  list
## [4,]    2   xml_node  list
## [5,]    2   xml_node  list
## [6,]    2   xml_node  list
```

```
length(all_titles)
```

```
## [1] 40
```

What do you notice?

TAKING A CLOSER LOOK

```
all_titles[1]
```

```
## {xml_nodeset (1)}
## [1] <h3 class="title">\n<a href="https://www.fbi.gov/wanted/kidnap/li:
```

It's the text of the `` tag.

1. GETTING A LIST OF ALL NAMES

1. Access the full html page `read_html(target_url)`
2. Search all h3 headings with class “title”
`html_nodes('h3.title')`
3. Find all `<a>` tags (= links) `html_nodes('a')`
4. Extract the text `html_text()`

COMBINED

```
all_names = target_page %>%
  html_nodes('h3.title') %>%
  html_nodes('a') %>%
  html_text()
```

```
all_names
```

```
## [1] "LISA MARIA SZASZ"  
## [2] "WILLIAM EBENEZER JONES, JR."  
## [3] "VANESSA MORALES"  
## [4] "FELIX BATISTA"  
## [5] "MARK HIMEBAUGH"  
## [6] "MICHAELA JOY GARECHT"  
## [7] "JANE MCDONALD-CRONE"  
## [8] "JALIEK L. RAINWALKER"  
## [9] "ARANZA MARIA OCHOA LOPEZ"  
## [10] "KARLIE LAIN GUSÉ"  
## [11] "KARLA RODRIGUEZ"  
## [12] "ENRIQUE RIOS"  
## [13] "ELIJAH MOORE"  
## [14] "LISA IRWIN"  
## [15] "DULCE MARIA ALAVEZ"  
## [16] "TARA LEIGH CALICO"  
## [17] "SHANNA GENELLE PEOPLES"  
## [18] "ARRY LYNN PATTERSON"
```

Getting all names: done!

2. STORING THE BIO INFORMATION

We know: there's a table with class
wanted-person-description that contains the data we
want.

But: we need to access each missing person!

For-loops to the rescue...

2. STORING THE BIO INFORMATION

1. Access *the full html page*
2. Search *all h3 headings with class “title”*
3. Find *all <a > tags (= links)*
4. Extract the ~~text~~ actual link
5. Access that page
6. Extract the table with class
wanted-person-description

GETTING THE LINK TO EACH PERSON

```
all_persons_links = target_page %>%
  html_nodes('h3.title') %>%
  html_nodes('a') %>%
  html_attr('href')
```

```
head(all_persons_links)
```

```
## [1] "https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz"  
## [2] "https://www.fbi.gov/wanted/kidnap/william-ebeneezer-jones-jr"  
## [3] "https://www.fbi.gov/wanted/kidnap/vanessa-morales"  
## [4] "https://www.fbi.gov/wanted/kidnap/felix-batista"  
## [5] "https://www.fbi.gov/wanted/kidnap/mark-himebaugh"  
## [6] "https://www.fbi.gov/wanted/kidnap/michaela-joy-garecht"
```

```
length(all_persons_links)
```

```
## [1] 40
```

2. STORING THE BIO INFORMATION

Before you write a loop...

```
lisa_maria = all_persons_links[1]
temp_target_url = lisa_maria
temp_target_page = read_html(temp_target_url)
```

SINGLE-CASE PROOF

```
description = temp_target_page %>%
  html_nodes('table.wanted-person-description') %>%
  html_table()

description
```

```
## [1]
##           X1          X2
## 1 Date(s) of Birth Used July 16, 1962
## 2 Place of Birth          Ohio
## 3           Hair        Black
## 4           Eyes       Hazel
## 5         Height      5 '7"
## 6        Weight    135 pounds
## 7           Sex     Female
## 8           Race      White
```

THE FOR-LOOP

1. do this for each link
2. store it somewhere (easiest: in a list)
3. log progress

```

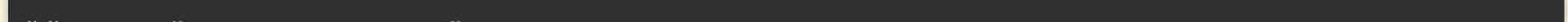
list_for_data = list()
for(i in all_persons_links){
  print(paste('Accessing:', i))
  temp_target_url = i
  temp_target_page = read_html(temp_target_url)
  description = temp_target_page %>%
    html_nodes('table.wanted-person-description') %>%
    html_table()
  index_of_i = which(i == all_persons_links)
  list_for_data[[index_of_i]] = description
  print('--- NEXT ---')
}

```

```

## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/william-ebeneezer-johnson"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/vanessa-morales"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/felix-batista"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/mark-himebaugh"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/michaela-joy-garecht"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/jane-mcdonald-crone"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/jalieka-l.-rainwalker"
## [1] "--- NEXT ---"
## [1] "Accessing: https://www.fbi.gov/wanted/kidnap/aranza-maria-ochoa"

```



Now we have a list of tables.

Each table contains the details of one missing person:

```
# thirteenth element in the list
list_for_data[[13]]
```

```
## [[1]]
##          x1                  x2
## 1 Date(s) of Birth Used      November 3, 1999
## 2                      Hair        Black
## 3                      Eyes       Brown
## 4                      Height     5 '11"
## 5                      Weight    200 pounds
## 6                      Sex        Male
## 7                      Race       Black
## 8 Scars and Marks Moore has a burn scar on his left hand.
```

3. DOWNLOADING THE POSTER

Each kidnapped person has a 'download link'...

<https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz/@download.pdf>

COMPARE THESE TWO

```
https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz/@download.pdf
```

```
lisa_maria
```

```
## [1] "https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz"
```

Notice something?

3. DOWNLOADING THE POSTER

We can just ‘work around’ this:

```
download_url_lisa_maria = paste(tolower(lisa_maria), '/@@download.pdf',  
download_url_lisa_maria
```

```
## [1] "https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz/@@download.pdf"
```

<https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz/@@download.pdf>

3. DOWNLOADING THE POSTER

Make use of R's vectorised structure:

```
all_download_links = paste(tolower(all_persons_links), '@@download.pdf'  
head(all_download_links)
```

```
## [1] "https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz/@@download.pdf"  
## [2] "https://www.fbi.gov/wanted/kidnap/william-ebeneezer-jones-jr/@@download.pdf"  
## [3] "https://www.fbi.gov/wanted/kidnap/vanessa-morales/@@download.pdf"  
## [4] "https://www.fbi.gov/wanted/kidnap/felix-batista/@@download.pdf"  
## [5] "https://www.fbi.gov/wanted/kidnap/mark-himebaugh/@@download.pdf"  
## [6] "https://www.fbi.gov/wanted/kidnap/michaela-joy-garecht/@@download.pdf"
```

Now:

1. access each
2. “download” (= write) the file
 - needs a filename on your computer

CREATE FILENAMES

```
library(stringr)
file_names = paste(all_names, '.pdf', sep="")
head(file_names)
```

```
## [1] "LISA MARIA SZASZ.pdf"          "WILLIAM EBENEZER JONES, JR.."
## [3] "VANESSA MORALES.pdf"            "FELIX BATISTA.pdf"
## [5] "MARK HIMEBAUGH.pdf"             "MICHAELA JOY GARECHT.pdf"
```

REFINE FILENAMES

```
refined_file_names = tolower(str_replace_all(string = file_names, pattern = c("\\.", "\-", "\_"), ""))  
head(refined_file_names)
```

```
## [1] "lisa_maria_szasz.pdf"          "william_ebeneezer_jones,_jr..pdf"  
## [3] "vanessa_morales.pdf"           "felix_batista.pdf"  
## [5] "mark_himebaugh.pdf"            "michaela_joy_garecht.pdf"
```

DOWNLOAD

Download each pdf from the url and use the refined filenames

```
for(i in 1:length(all_download_links)){
  print(paste('Accessing URL: ', all_download_links[i], sep=""))
  download.file(url = all_download_links[i]
                 , destfile = refined_file_names[i]
                 , mode = "wb")
}
```

```
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/lisa-maria-szasz"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/william-ebeneeze"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/vanessa-morales"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/felix-batista/@0"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/mark-himebaugh/0"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/michaela-joy-gardner"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/jane-mcdonald-clark"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/jalieka-l.-rainwater"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/aranza-maria-ocampo"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/karlie-lain-gusewski"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/karla-rodriguez"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/enrique-rios/@@@"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/elijah-moore/@@@"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/lisa-irwin/@@down"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/dulce-maria-alava"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/tara-leigh-calio"
## [1] "Accessing URL: https://www.fbi.gov/wanted/kidnap/shanna-genelle-palmer"
```

STATIC VS DYNAMIC WEB-SCRAPING

What is dynamic content?

```
setTimeout(function(){
  alert("This is a delayed alert");
}, 4000);
```

SCRAPING DYNAMIC WEBPAGES

- problem for the snapshot method
- what if content loads after, say, 5 seconds?
- or if you can only send a request every 5 seconds?

SIMULATING TIMEOUTS

1. we need a way to simulate a browser
2. we need to simulate human user interaction

Enter: **RSelenium**

SETUP

```
library(RSelenium)

#make a connection
selenium_firefox = rsDriver(browser=c("firefox"))

#start a driver
driver = selenium_firefox$client
```

Live demo

BACKUP

```
#set target url
target_url = 'https://www.fbi.gov/wanted/kidnap'

#navigate the driver (= simulated browser) to the target url
driver$navigate(target_url)
```

BACKUP (1)

```
# 1. set wait intervals
list_for_requests = list()

for(i in 1:5){
  parsed_pagesource <- driver$getPageSource()[[1]]
  result <- read_html(parsed_pagesource) %>%
    html_nodes('h3.title') %>%
    html_nodes('a')

  list_for_requests[[i]] = result
  print(paste('Sent request at:', Sys.time(), sep=" "))

  Sys.sleep(5)
}
```

BACKUP (2)

```
# 2. simulate scroll
#navigate the driver (= simulated browser) to the target url
driver$navigate(target_url)

#find the html body
page_body = driver$findElement("css", "body")

#send a scroll command (note that this is a page_down request in Javascript)
page_body$sendKeysToElement(list(key = "page_down"))
```

BACKUP (3)

```
# 3. simulate multiple scrolls
#navigate the driver (= simulated browser) to the target url
driver$navigate(target_url)

#find the html body
page_body = driver$findElement("css", "body")

#send multiple scroll commands in a loop
for(i in 1:10){
  page_body$sendKeysToElement(list("key"="page_down"))

  # allow some time for this to happen (here: 3 seconds)
  Sys.sleep(3)
}
```

BACKUP (3 CONT'D)

```
#now access the page source (important: you need to do this through the driver)
parsed_pagesource <- driver$getPageSource()[[1]]  
  
#now we can scrape from the page after the simulation
full_results <- read_html(parsed_pagesource) %>%
  html_nodes('h3.title') %>%
  html_nodes('a') %>%
  html_attr('href')  
  
length(full_results)
```

BACKUP (CLOSE)

```
# close the driver and the server
driver$close()

selenium_firefox$server$stop()
```

NOTES ON WEBSRAPING

- highly customisable (= juicy data)
- basically: “anything goes”
- can be unstable/sensitive to html changes

SAME IDEA, DIFFERENT HTML

Hi! Sign in or register | Daily Deals | Gift Cards | Help & Contact

Sell | My eBay |

TV receiver | Satellite TV Receivers | Search

Related: digital tv receiver satellite tv receiver digital tv converter box tv receiver for android tv...

All Listings | Accepts Offers | Auction | Buy It Now | Best Match |

2,096 results | Save this search

Price: Under \$20.00 | \$20.00 - \$30.00 | Over \$30.00

Freesat V7S HD FTA Digital Satellite TV Receiver DVB-S2/S Support BissKey 1080P
Brand New
\$22.99 to \$24.99 From China
Buy It Now | Free International Shipping

Support PowerVu+Bliss Key+Wifi

Digital DVB-S2 Satellite Receiver Converter Tuner Wifi Combo Youtube FTA Tv Box~
Brand New
\$11.58 From China
Buy It Now | Free International Shipping | 3 Watching

CCcam Server 1 year HD Satellite TV

Feedback

Inspector | Console | Debugger | Style Editor | ... | Search HTML

<!DOCTYPE html>
<!--[if IE 9]>html class="ie9" lang="en"><![endif]-->
<!--[if gt IE 9]><!-->
<html class="history devicemotion deviceorientation gr__ebay_com" lang="en"> event
<!-->[endif]-->
<head></head>
<body class="s-page no-touch skin-large srp--list-view no-touch skin-large gh-l199 gh-979 gh-939 gh-899 gh-799 gh-flex" data-gr-c-s-loaded="true" style="background-image: url('data:image/png;base64,iVBORw0KGgoAAA...eat: repeat-x, repeat; background-position: 0px 30px, 0% 0%'> event
<div id="gh-gb" tabindex="-1"></div>
<div class="x-header"></div>
<script></script>
<script></script>
<noscript id="w1"></noscript>
<script></script>
<script></script>
<div class="srp-main srp-main--isLarge"></div>
<div class="x-footer"></div>
<div id="w14" class="hide"></div>
<div id="w15" class="srp-mask"></div> event
<div class="s-modal-wrapper" style="position: relative;"></div>
<script></script>
<script></script>
<script src="https://ir.ebaystatic.com/rs/c/inception-1140e9.js"></script>
<script src="https://ir.ebaystatic.com/rs/c/search-page-large-20190108195210-3ea83c.js"></script>
<style type="text/css"></style>
<div id="lens-modal-wrapper0" class="lens-modal-wrapper" style="z-index: 10100030;"></div>
<script>\$.mod.ready();</script>
<script type="text/javascript" src="https://ir.ebaystatic.com/rs/pj0rx2hna0xlfuril1xhghtrab.js"></script>
<script type="text/javascript" src="https://ir.ebaystatic.com/rs/c/makeebayfasterscript-src-scripts-body-78a2168a.js"></script>
<script type="text/javascript"></script>
<script></script>
<script type="text/javascript"></script>
<script type="text/javascript"></script>
<script id="taasHeaderRes" type="text/javascript" src="https://ir.ebaystatic.com/rs/u/10341vh50v721mhhu due1dm5wad_ie"></script>
html.history.devicemotion.deviceorientat... > body.s-page.no-touch.skin-large.srp--lis...
Filter Styles | + .cls | Layout | Computed | Animations | Fonts
Pseudo-elements | Filter Styles | Browser styles
This Element | background-color | rgb(247, 247, 247)

SAME IDEA, DIFFERENT HTML

SAME IDEA, DIFFERENT HTML

The screenshot shows the Trustpilot review page for Vodafone. The main content includes:

- A screenshot of the Vodafone website.
- The company name "Vodafone".
- The number of reviews "Reviews 7,048" and the rating "Bad".
- A star rating icon showing 1 star out of 5.
- A link to the website "www.vodafone.co.uk".
- A status indicator "Claimed" with a checkmark icon.
- A "Write a review" button with a user icon.
- A 5-star rating scale.
- A review distribution chart:

Rating Category	Percentage
Excellent	8%
Great	4%
Average	2%
Poor	3%
Bad	84%

- A user profile for "ashley ounsworth" with "1 review".

The right side of the image shows the browser's developer tools (Inspector tab selected) with the following details:

- Search bar: Search HTML
- HTML code snippet (highlighted):

```
> <div class="chart__cell chart__cell_value--2-of-5">
>   <div class="chart__row star-rating-2" title="195 of 7,048 reviews">@@</div>
>   <div class="chart__row star-rating-1" title="5,928 of 7,048 reviews">@@</div>
> </div>
> </div>
> <div class="review-list" data-review-list="">
>   <script type="text/javascript">@@</script>
>   <div class="review-card ">@@</div>
>   <div class="ad-block ">@@</div>
>   <div class="review-card ">@@</div>
>   <div class="ad-block ">@@</div>
>   <div class="review-card ">@@</div>
>   <div class="ad-block ">@@</div>
>   <div class="review-card review-card--has-stack">@@</div>
>   <div class="review-card ">@@</div>
> </div>
```
- Element path: <div> > section.reviews-container > div.review-list > div.review-card. > article#5c44c77a9d378009a45f776f.review
- Inspector tabs: Filter Styles, .cls, Layout, Computed, Animations, Fonts (Layout tab selected).

SAME IDEA, DIFFERENT HTML

Phone: (NA) -NA
Email: Email Seller
Location: Michigan
Website: NA

We have available one tame hand raised baby female Kinkajou, perfect as an educational ambassador or pet. We are USDA licensed, serious and educated inquiries only please.

[View Details](#)

A photograph showing two adult Kinkajous (coati-like mammals) sitting together in a wire cage. One is facing forward, the other is behind it. They have brown fur and large eyes.

Adult Kinkajous

Name: CJG EXOTICS [View Profile](#)
Posted: 1/20/2019
Phone: (NA) -NA
Email: Email Seller
Location: Michigan
Website: NA

We have available several unrelated adult Kinkajou pairs and trios. These would make an excellent breeding project or zoo exhibit. These are not tame and not pets, we are USDA licensed.

[View Details](#)

A screenshot of a browser's developer tools, specifically the Elements tab. It shows the entire HTML code structure of the page, including numerous rows of repeating div elements and styling rules like padding-top: -25px !important. The code is heavily nested, indicating a complex web design. At the bottom of the code, there are sections for 'Styles', 'Event Listeners', 'DOM Breakpoints', 'Properties', and 'Accessibility'.

RECAP

- Always: problem first, never the method first!
- Method follows problem!
- HTML structure key to webscraping
- Webscraping:
 - understanding the structure of a webpage
 - exploiting that structure for web-scraping
- principle is always the same: understand + exploit the html structure

WHAT'S NEXT?

- Today's tutorial: dynamic scraping of the FBI's website, full pipeline for exotic animal trading forum
- Homework: Rvest tutorials, webscraping practice

Next week: Text Mining 1