Applied Web Scraping

Tobias Rüttenauer
2023-03-27

Introduction

Course

https://github.com/ruettenauer/Web_Scraping

This course profited a lot from existing materials by

- Data science for economists by Grant McDermott
- Bit by Bit by Matt Salganic
- Digital Trace Data by Ridhi Kashyap
- Webscraping Introduction by Theresa Gessler
- Computational Social Science by Paul Bauer

Me

Tobias Rüttenauer

t.ruttenauer@ucl.ac.uk

Lecturer in Quantitative Social Science (Environmental Sociology)

Digital Trace Data

Social transformation:

- Digital technologies permeate social, economic, political life.
- A lot of social interactions happen in the digital world.

Data revolution:

- The digitalisation of our lives creates data by-products: digital trace data.
- Data that require **repurposing** because they were not intentionally collected for research.
- Existing data may require a lot of thoughts and work to fit your purpose.

Research revolution:

- Ask the right questions!
- Huge amounts of (big) data, but: **What things are worth counting?**

Digital Trace Data

Social research data in a digital age

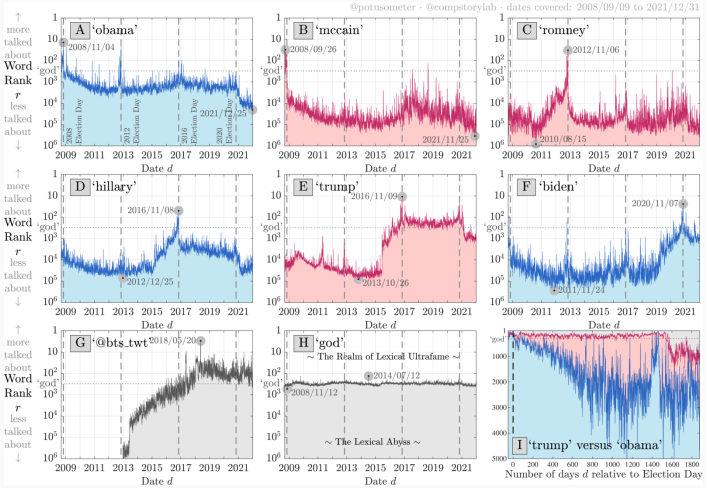
Readymade

Custommade

Digital Trace Data

Strenghs of Trace Data

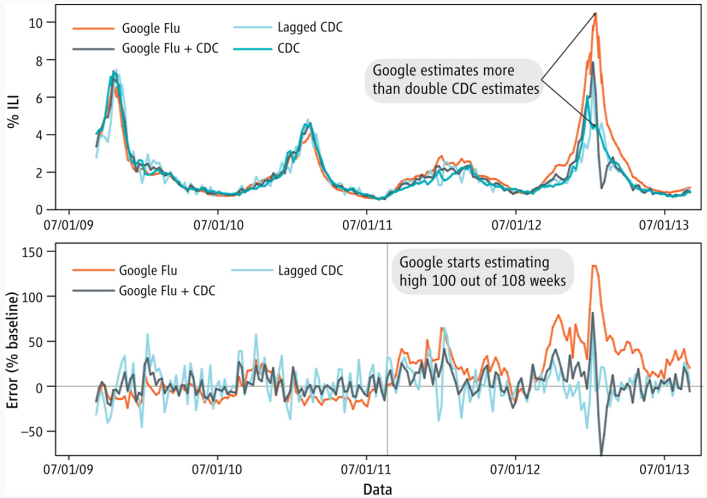
- Big
- Always on
- Actual behaviour
- Non-reactive
- Captures social relations
- Mergable (e.g. geo-coordinates)



Dodds et al, ArXiv

Weaknesses of Trace Data

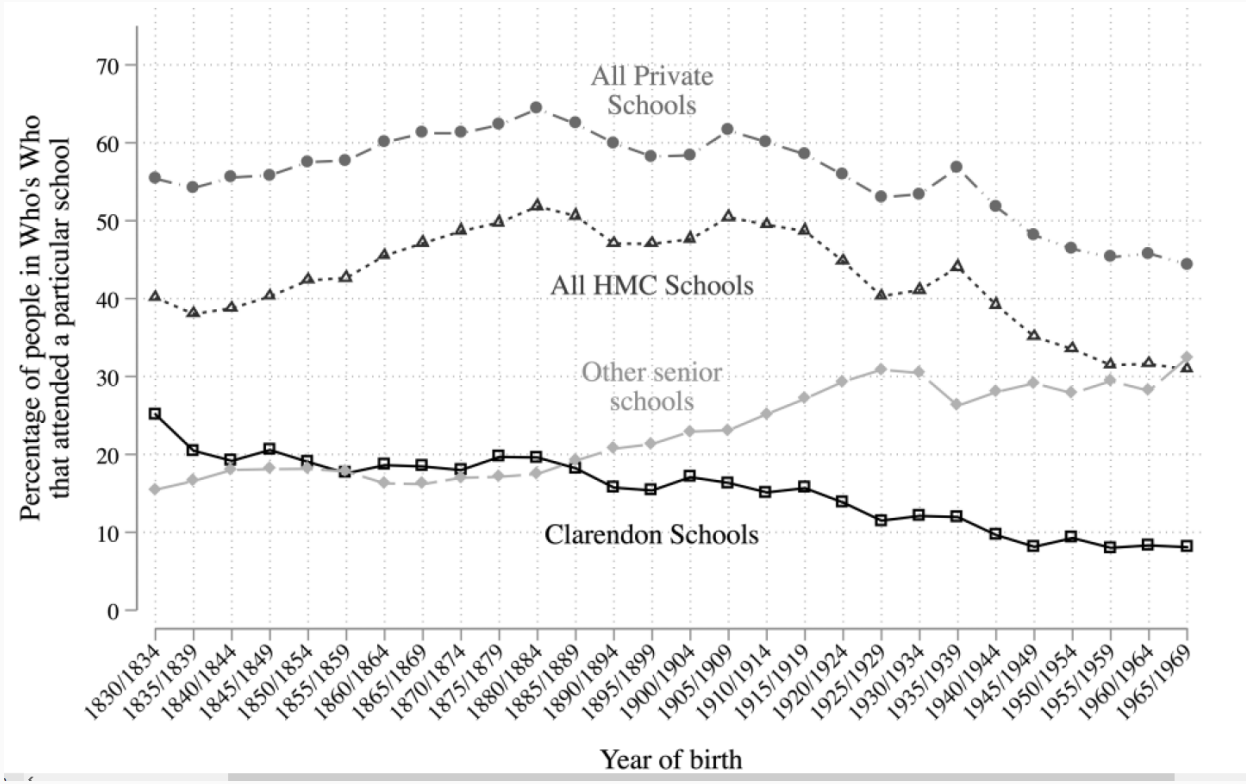
- Incomplete & Inaccessible
- Non-representative & Selective
- Drifting
- Algorithmically confounded
- Dirty
- Sensitive to privacy issues



Lazer et al, Science

Examples

The Decline and Persistence of the Old Boy: Private Schools and Elite Recruitment

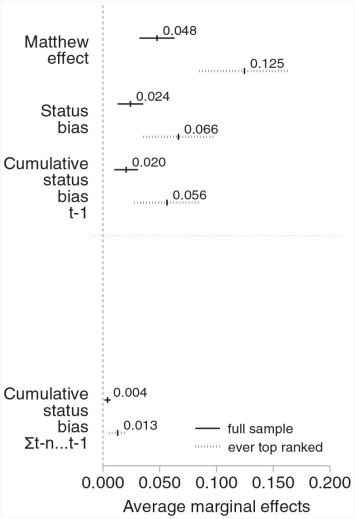


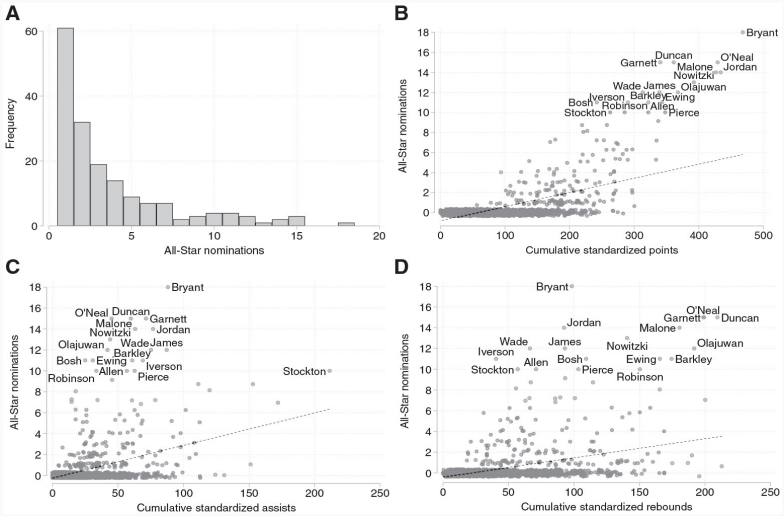
Reeves et al., American Sociological Review

Examples

Matthew Effect & Cumulative Status Bias in NBA All-Star Elections

- based on 1.2 million game logs from the 3,300 players (www.basketball-reference.com/)

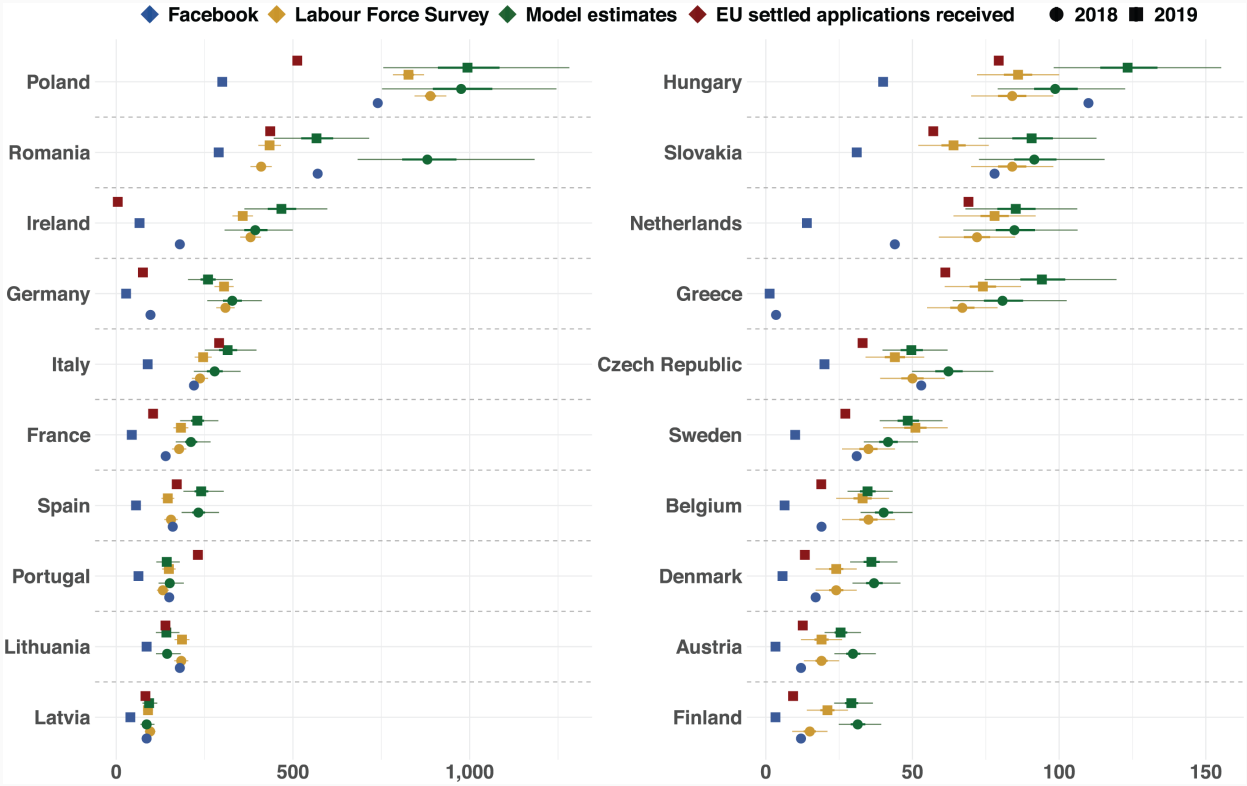




Biegert et al., American Sociological Review

Examples

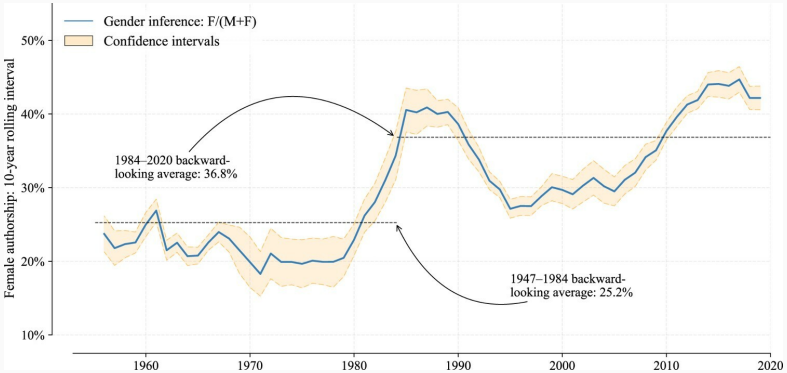
Estimating Migrant Stocks Using Digital Traces and Survey Data

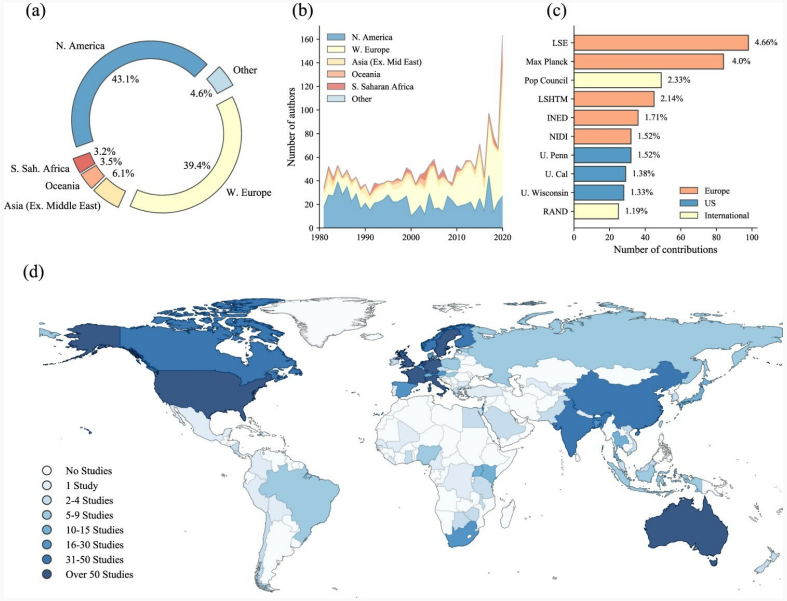


Rampazzo et al., Demography

Examples

Population Studies: Authors and topics over the past 75 years





Mills & Rahal., Population Studies

Examples

Increase in Discrimination Against Arabs and Muslims on AirBnB after Paris Attacks

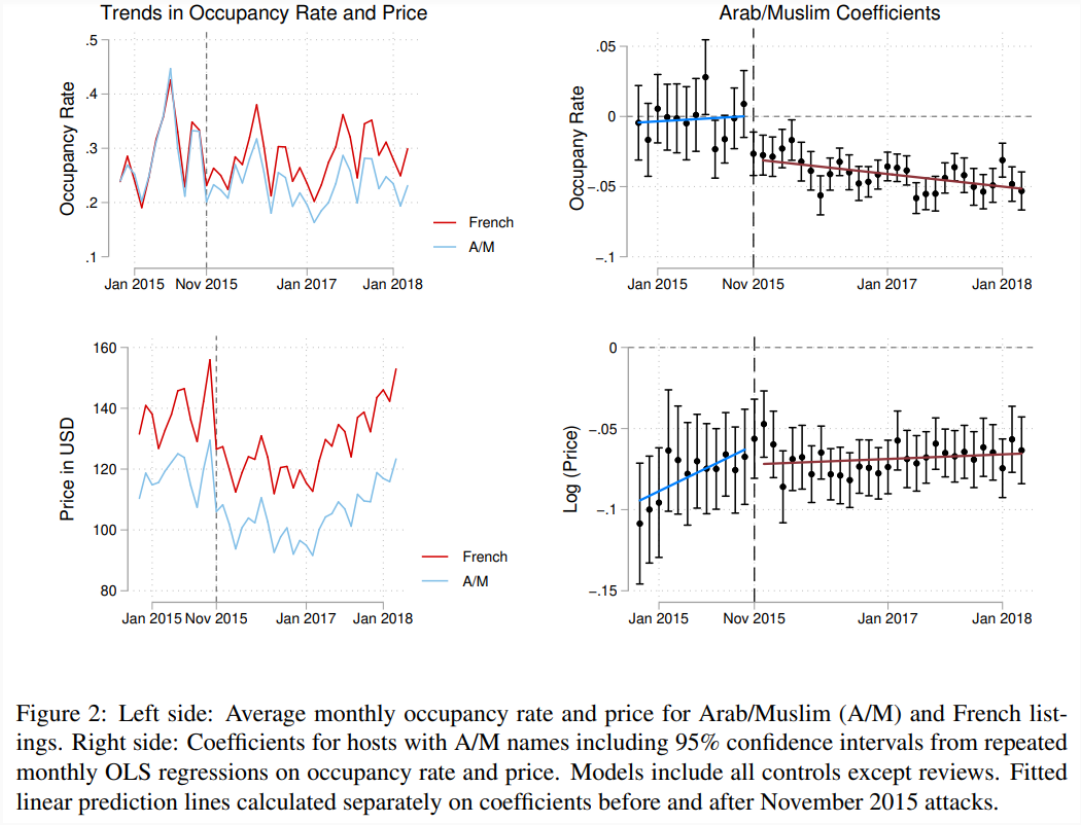


Figure 2: Left side: Average monthly occupancy rate and price for Arab/Muslim (A/M) and French listings. Right side: Coefficients for hosts with A/M names including 95% confidence intervals from repeated monthly OLS regressions on occupancy rate and price. Models include all controls except reviews. Fitted linear prediction lines calculated separately on coefficients before and after November 2015 attacks.

Accessing Digital Trace Data

1) Web (Screen) scraping

2) Data from application programming interfaces (APIs)

3) Archives or data sharing platforms

- e.g. Google Trends, IPUMS, Our World in Data, London Datastore

4) Data sharing agreements with data owners

- e.g. UK Data Service, or commercial platforms

Web Scraping

What is webscraping?

Extracting data from webpages

- Anything from university webpage and geographic information to social media
- Lots of different techniques
- Process unstructured (messy) data and make structured (data.table)
- **~ Automated copy & paste**

Types of scraping

- Gathering as diverse information as possible from different pages vs.
- Very specific scrapers
- Fully automated scrapers to half-automated scripts
- Single-use scraping vs. regular data collection

Sofware for scraping

- Python
- R & Rstudio

Current debates about scraping

Is scraping legal?

- Read the website's Terms of Service: are you allowed to do this?
- Larger websites like Facebook, Instagram, NY Times do not allow these practices
- However, a recent LinkedIn court case in th US allows webscraping
- Just because you can scrape it, doesn't mean you should

Is scraping ethical?

- are the data sensitive?
- Could the use of the data harm in some way?
- freely accessible content vs. commercial content

The rules of the game

Respect the hosting site's wishes

- Check if an API exists or if data are available for download
- Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
- Some websites disallow scrapers on robots.txt file

Limit your bandwidth use

- Wait one or two seconds after each hit
- Scrape only what you need, and just once (e.g. store the file on disk, and then parse it)
- ...otherwise you'll get a visit from the IT guys! (e.g. scraping articles)

Respect data protection

- Secure storage vs. deletion of data
- Anonymization of users

Browsing vs. scraping

Browsing

- You click on something
- Browser sends request to server that hosts webpage
- Server returns resource (e.g. HTML document)
- Browser interprets HTML and renders it in a nice fashion

Scraping with R

- You manually specify a resource
- R sends request to server that hosts website
- Server returns resource
- R parses HTML (i.e., interprets the structure), but does not render it in a nice fashion
- You tell R which parts of the structure to focus on and what to extract

First step is to **understand some HTML**

HTML: a primer

HTML is text with marked-up structure, defined by **tags**:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Some common tags:

- Document elements: `<head>`, `<body>`, `<footer>`...
- Document components: `<title>`, `<h1>`, `<p>`, `<div>`...
- Text style: `<b>`, `<i>`, `<strong>`...
- Hyperlinks: `<a href="url">`

General structure

- `<tagname>Content goes here ... </tagname>`

Inspect your website

CSS

- Cascading Style Sheets (CSS): describes formatting of HTML components
- *Properties*. CSS properties are the “how” of the display rules. These are things like which font family, styles and colours to use, page width, etc.
- *Selectors*. CSS selectors are the “what” of the display rules. They identify which rules should be applied to which elements. E.g. Text elements that are selected as “.h1” (i.e. top line headers) are usually larger and displayed more prominently than text elements selected as “.h2” (i.e. sub-headers).
- The key point is that if you can identify the CSS selector(s) of the content you want, then you can isolate it from the rest of the webpage content that you don’t want. This where SelectorGadget comes in.

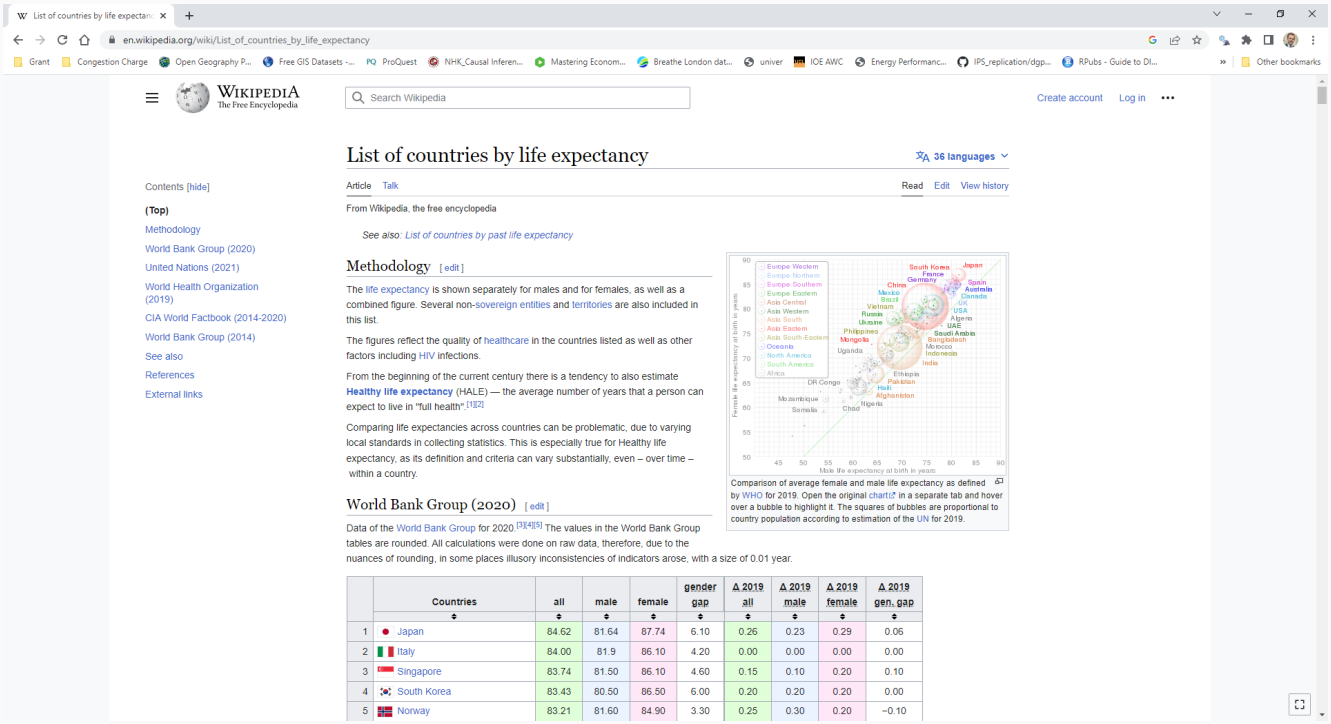
SelectorGadget

- SelectorGadget as Chrome Extension
- Inspect option in Chrome

Inspect your website

Wikipedia Example

- https://en.wikipedia.org/wiki/List_of_countries_by_life_expectancy



List of countries by life expectancy

Contents [hide]

(Top)
Methodology
World Bank Group (2020)
United Nations (2021)
World Health Organization (2019)
CIA World Factbook (2014-2020)
World Bank Group (2014)
See also
References
External links

From Wikipedia, the free encyclopedia

See also: List of countries by past life expectancy

Methodology [edit]

The life expectancy is shown separately for males and for females, as well as a combined figure. Several non-sovereign entities and territories are also included in this list.

The figures reflect the quality of healthcare in the countries listed as well as other factors including HIV infections.

From the beginning of the current century there is a tendency to also estimate **Healthy life expectancy** (HALE) — the average number of years that a person can expect to live in "full health".[1][2]

Comparing life expectancies across countries can be problematic, due to varying local standards in collecting statistics. This is especially true for Healthy life expectancy, as its definition and criteria can vary substantially, even – over time – within a country.

Comparison of average female and male life expectancy as defined by WHO for 2019. Open the original chart in a separate tab and hover over a bubble to highlight it. The squares of bubbles are proportional to country population according to estimation of the UN for 2019.

World Bank Group (2020) [edit]

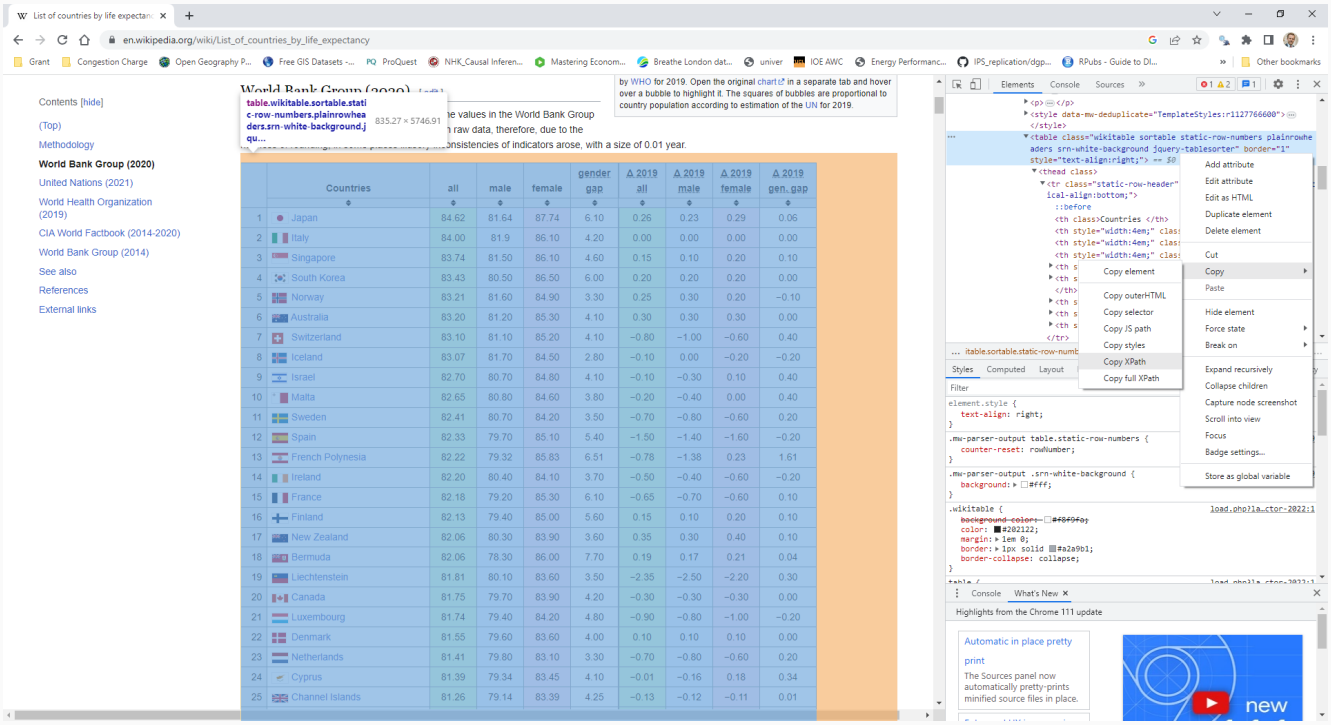
Data of the World Bank Group for 2020.[3][4][5] The values in the World Bank Group tables are rounded. All calculations were done on raw data, therefore, due to the nuances of rounding, in some places illusory inconsistencies of indicators arose, with a size of 0.01 year.

	Countries	all	male	female	gender gap	Δ 2019 all	Δ 2019 male	Δ 2019 female	Δ 2019 gen. gap
1	Japan	84.62	81.64	87.74	6.10	0.26	0.23	0.29	0.06
2	Italy	84.00	81.9	86.10	4.20	0.00	0.00	0.00	0.00
3	Singapore	83.74	81.50	86.10	4.60	0.15	0.10	0.20	0.10
4	South Korea	83.43	80.50	86.50	6.00	0.20	0.20	0.20	0.00
5	Norway	83.21	81.60	84.90	3.30	0.25	0.30	0.20	−0.10

Inspect your website

Wikipedia Example

- Use inspect option to select table
- copy Xpath
- In this case: `//*[@id="mw-content-text"]/div[1]/table[1]`



Scrape the website

R Package `rvest`

- Package for parsing and scraping
- Covers most frequent use cases
- but relatively simple: no dynamic webpages

rvest Vignette

Some important commands

- `read_html()`
- `html_nodes()`
- `html_elements()`
- `html_text2()`
- `html_table()`
- `html_attrs()`

Example 1: Wiki

Parsing the url of the website

```
library(rvest)
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
library(xml2)
url ← "https://en.wikipedia.org/wiki/List_of_countries_by_life_expectancy"
parsed ← read_html(url)
```

This returns an xml object that contains all the information of the website.

```
parsed
```

```
## {html_document}
## <html class="client-nojs vector-feature-language-in-header-enabled vector-feature-langua
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ..
## [2] <body class="skin-vector skin-vector-search-vue mediawiki ltr sitedir-ltr
```

Example 1: Wiki

Extract the information we need

- we use the xpath that we have extracted above

```
# Select the desired element
parsed.sub ← html_element(parsed, xpath = '//*[@id="mw-content-text"]/div[1]/table[1]

# Convert to table
table1.df ← html_table(parsed.sub)

head(table1.df)
```

```
## # A tibble: 6 × 9
##   Countries       all  male female gendergap `Δ 2019all` Δ 201…¹ Δ 201…² Δ 201…³
##   <chr>         <dbl> <dbl>  <dbl>     <dbl> <chr>       <chr>   <chr>   <chr>
## 1 ""             NA    NA     NA        NA   ""          ""      ""      ""
## 2 "Japan"        84.6  81.6   87.7       6.1 "0.26"      "0.23"  "0.29"  "0.06"
## 3 "Italy"        84    81.9   86.1       4.2 "0.00"      "0.00"  "0.00"  "0.00"
## 4 "Singapore"    83.7  81.5   86.1       4.6 "0.15"      "0.10"  "0.20"  "0.10"
## 5 "South Korea"  83.4  80.5   86.5       6   "0.20"      "0.20"  "0.20"  "0.00"
## 6 "Norway"       83.2  81.6   84.9       3.3 "0.25"      "0.30"  "0.20"  "-0.10"
## # … with abbreviated variable names ¹`Δ 2019male`, ²`Δ 2019female`,
## #   ³`Δ 2019gen. gap`
```

Example 1: Wiki

Some cleaning

- it always makes sense to clean the columns names

```
library(janitor)

# clean names
names(table1.df) <-  janitor::make_clean_names(names(table1.df))

# Delete empty rows
empt <- apply(table1.df, 1, FUN = function(x) all(is.na(x) | x == ""))
table1.df <- table1.df[which(!empt), ]

head(table1.df)
```

```
## # A tibble: 6 × 9
##   countries     all  male female gendergap d_2019all d_2019male d_2019…¹ d_201…²
##   <chr>       <dbl> <dbl>  <dbl>     <dbl> <chr>     <chr>      <chr>    <chr>
## 1 Japan        84.6  81.6   87.7       6.1 0.26      0.23       0.29     0.06
## 2 Italy        84    81.9   86.1       4.2 0.00      0.00       0.00     0.00
## 3 Singapore    83.7  81.5   86.1       4.6 0.15      0.10       0.20     0.10
## 4 South Korea  83.4  80.5   86.5       6   0.20      0.20       0.20     0.00
## 5 Norway       83.2  81.6   84.9       3.3 0.25      0.30       0.20     -0.10
## 6 Australia    83.2  81.2   85.3       4.1 0.30      0.30       0.30     0.00
```

Example 2: Starwars

Wiki Example

- Wikipedia pages have a specific structure
- Data is stored in tables
- We can extract these tables

Real world

- Websites are often "unstructured"
- See for instance https://rvest.tidyverse.org/articles/starwars.html
- We need to use some generic selectors to structure the data

Example 2: Starwars

```
starwars ← read_html("https://rvest.tidyverse.org/articles/starwars.html")

# Extracting each section
films ← html_elements(starwars, "section")

films
```

```
## {xml_nodeset (7)}
## [1] <section><h2 data-id="1">\nThe Phantom Menace\n</h2>\n<p>\nReleased: 1999 ...
## [2] <section><h2 data-id="2">\nAttack of the Clones\n</h2>\n<p>\nReleased: 20 ...
## [3] <section><h2 data-id="3">\nRevenge of the Sith\n</h2>\n<p>\nReleased: 200 ...
## [4] <section><h2 data-id="4">\nA New Hope\n</h2>\n<p>\nReleased: 1977-05-25\n ...
## [5] <section><h2 data-id="5">\nThe Empire Strikes Back\n</h2>\n<p>\nReleased: ...
## [6] <section><h2 data-id="6">\nReturn of the Jedi\n</h2>\n<p>\nReleased: 1983 ...
## [7] <section><h2 data-id="7">\nThe Force Awakens\n</h2>\n<p>\nReleased: 2015- ...
```

```
# We have a structure now, but we need to extract the elements

# Extract titles (which are all headings order 2 → h2)
titles ← html_elements(films, "h2")
titles ← html_text2(titles) # Transform into plain text
```

Example 2: Starwars

- Extract titles and dates using CSS rules
- Use regular expressions to select correct lines

```
# Extract dates
dates ← html_text2(html_elements(films, "p"))
dates ← dates[grepl("^Released", dates)] # Subset to lines beginning with "Released"

# Alternatively we can select the paragraph after heading 2
dates ← html_text2(html_elements(films, "h2 + p"))
dates ← gsub("^Released: ", "", dates) # clean text

head(data.frame(titles, dates))
```

```
##                     titles      dates
## 1      The Phantom Menace 1999-05-19
## 2    Attack of the Clones 2002-05-16
## 3     Revenge of the Sith 2005-05-19
## 4               A New Hope 1977-05-25
## 5 The Empire Strikes Back 1980-05-17
## 6      Return of the Jedi 1983-05-25
```

CSS selector rules

Simple rules

Basic selector	Kind	Description
*	Universal selector	Matches all elements
name	Type selector	For any given name, matches all elements with that tag name.
.name	Class selector	Matches all elements whose class attribute includes the word name.
#name	ID selector	Matches all elements whose id attribute is exactly name.
[attr]	Attribute selector	Matches all elements having an attribute named attr.
[attr=value]	Attribute selector	Matches all elements having an attribute named attr whose value is value.
[attr="value"]	Attribute selector	Same. With quotes around value, it may contain spaces and punctuation.
:nth-child(n)	Pseudo-class	Matches all elements that are child number n (ignoring text and attribute nodes) under their parent.
:nth-of-type(n)	Pseudo-class	Matches all elements that are child number n of that same tag name under their parent.

CSS selector rules

Rule combinations

Selector combination	Name	Description
C > T	Child	Only match T when it is a child of an element matched by C.
C T	Descendant	Only match T when it is a descendant of an element matched by C.
C + T	Adjacent sibling	Only match T when it immediately follows C as C's sibling.
C ~ T	Sibling	Only match T when it is a sibling of C (regardless of order).

See more CSS rules here

Why scraping?

Why scraping? I could just copy and paste this!

for() loops

for() loops let you

- run the same code along a series of indices `i`
- run the same code across the elements of a vector (e.g. names / dates / urls)
- repeat the same (scraping) task for multiple pages in seconds

```
for(i in c(1:3)){
  pi ← paste0("https://nonsense/p", i)
  print(pi)
  Sys.sleep(2) # Sleep for 2 seconds before next task
}
```

```
## [1] "https://nonsense/p1"
## [1] "https://nonsense/p2"
## [1] "https://nonsense/p3"
```

- If you scrape inside a loop, **don't overwhelm the server side!**
- use `Sys.sleep()` to **pause between tasks!**

Web Scraping Summary

Extracting data from webpages

- Anything from university webpage and geographic information to social media
- Lots of different techniques
- Huge amounts of (big) data, Ask the right questions!
- Respect the rules of the game

Limits

- Simple examples. Often web pages are complex with many different elements.
- Yielding messy data.
- Web / Screen scraping can be frustrating or unfeasible. What then?
  - For complex web pages, crowd worker platforms (e.g. Mechanical Turk) could be an option.
  - Some web data can be accessed via APIs.

APIs

APIs

What is an API?

- Application programming interfaces or APIs are a software intermediary that allows two applications to talk to each other.
- Web APIs allow one computer (a client) to ask another computer (a server) for some resource over the internet.
- APIs provide a structured way to access data that are stored in databases that are continuously updated.

Advatages

- Modern APIs adhere to standards, that make data exchange programmatically accessible, safe and structured
- Contrast with web scraping.
- Two important concepts when using APIs
    - Credentialling
    - Rate limiting

APIs - how it works

- Let's get some World Bank Data

1) Search for indicators you want

```
library(WDI)

# Search GDP per capita
WDIsearch("GDP.*capita.*constant")
```

```
##                   indicator                                                 name
## 717     6.0.GDPpc_constant GDP per capita, PPP (constant 2011 international $)
## 11431       NY.GDP.PCAP.KD                    GDP per capita (constant 2015 US$)
## 11433       NY.GDP.PCAP.KN                         GDP per capita (constant LCU)
## 11435    NY.GDP.PCAP.PP.KD  GDP per capita, PPP (constant 2017 international $)
## 11436 NY.GDP.PCAP.PP.KD.87  GDP per capita, PPP (constant 1987 international $)
```

```
# Search GDP per capita
WDIsearch("CO2.*capita")
```

```
##           indicator
## 6032 EN.ATM.CO2E.PC
## 6048 EN.ATM.METH.PC
## 6059 EN.ATM.NOXE.PC
##                                                                    name
```

APIs - how it works

2) Query the indicators

```
# Define countries, indicators form above, and time period
wd.df ← WDI(country = "all",
            indicator = c('population' = "SP.POP.TOTL",
                          'gdp_pc' = "NY.GDP.PCAP.KD",
                          'co2_pc' = "EN.ATM.CO2E.PC"),
            extra = TRUE,
            start = 2019, end = 2019)
```

3) Clean the data if necessary and plot

```
# Drop all country aggregrates
wd.df ← wd.df[which(wd.df$region ≠ "Aggregates"), ]
```

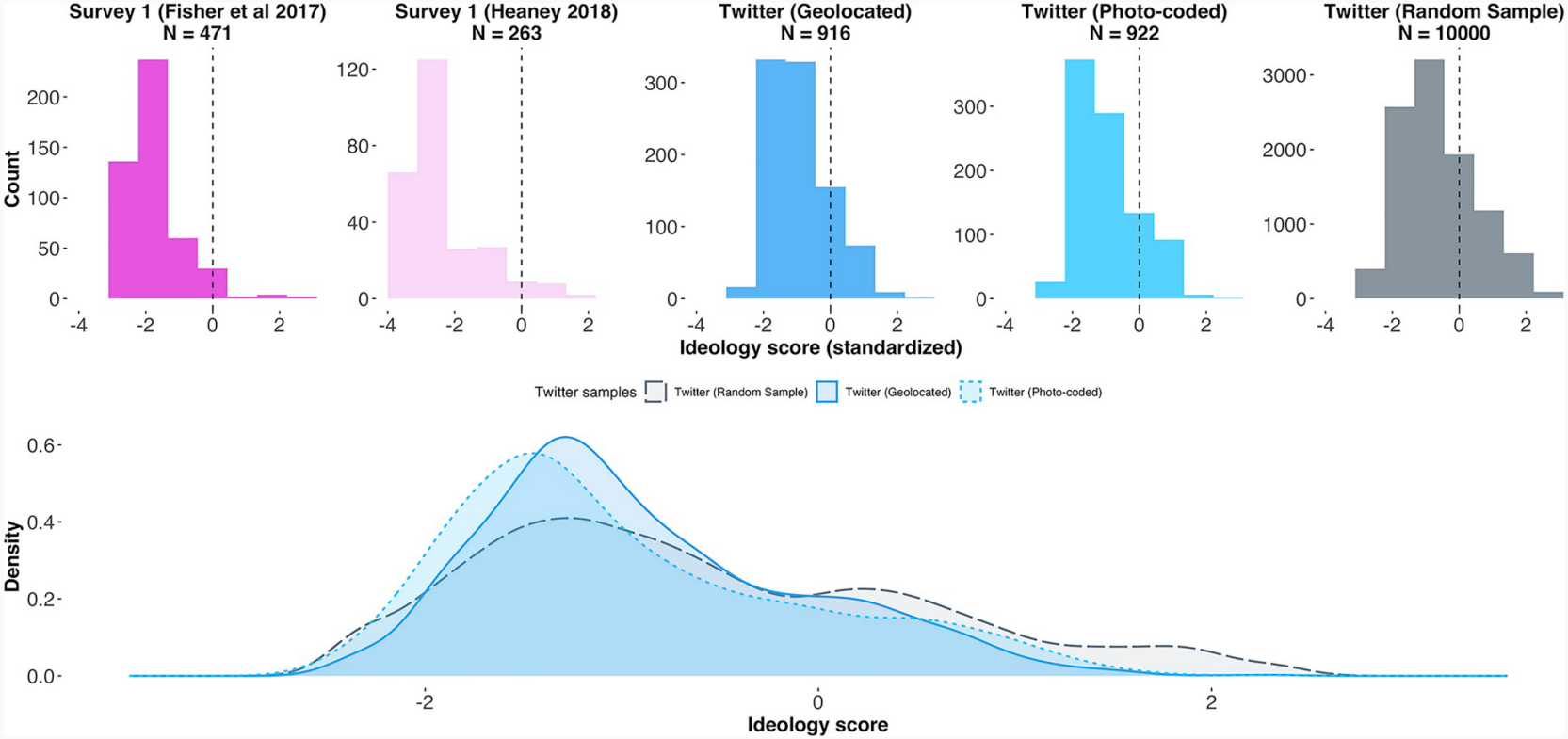
```
library(ggplot2)
pl ← ggplot(wd.df, aes(x = gdp_pc, y = co2_pc, size = population, color = region)) +
  geom_smooth(aes(group = 1), show.legend = "none") + geom_point(alpha = 0.5) +
  theme_minimal() + scale_y_log10() +  scale_x_log10(labels = scales::dollar_format()) +
  labs(y = "CO2 emissions per capita", x = "GDP per capita")
```

The Environmental Kuznet "curve"

```
pl
```

APIs - other examples

Twitter API - Ideologies of protesters

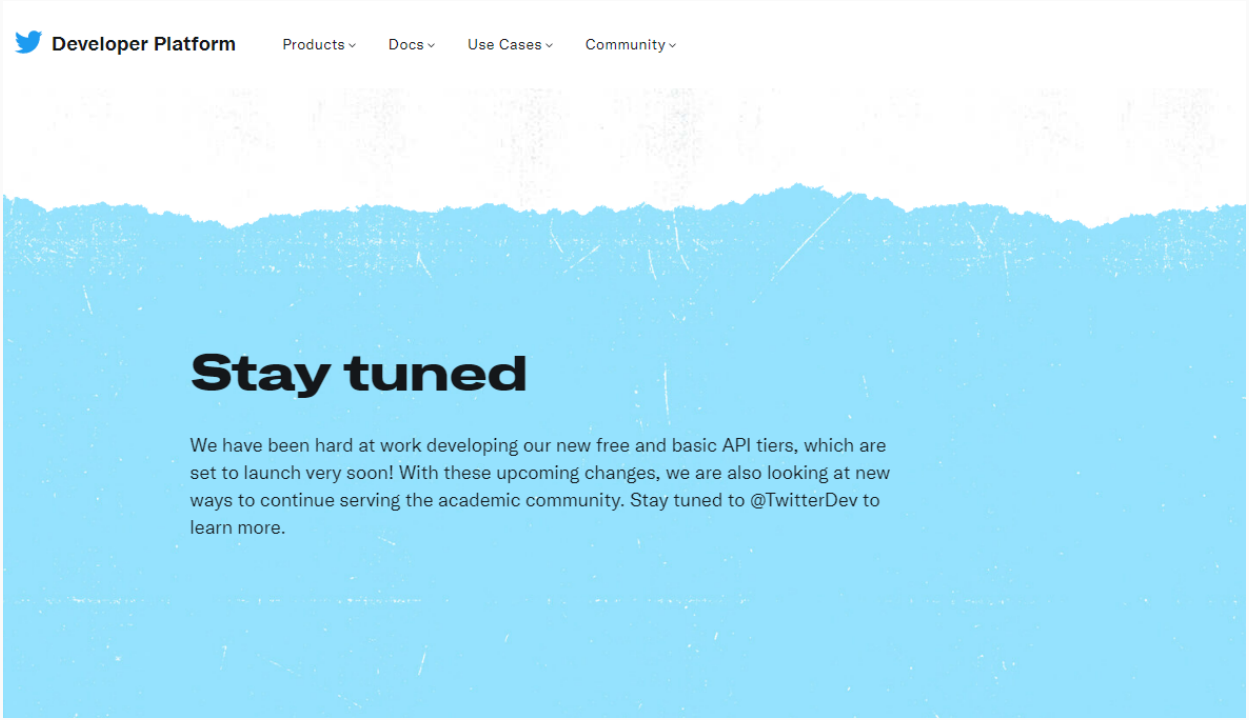


Barrie and Frey, PLOS One

APIs - other examples

Twitter API

See academictwitteR for Twitter API



APIs - other examples

Post-API Age ?

- Companies restrict access to their data
- Companies share only selective data

However, the availability of data still incraeses

- See a huge collection of social media APIs, including:
  - Facebook Ads
  - Google Trends
  - Reddit
  - Open Street Map
- More and more high-qaulity administrative data, e.g. IPUMS, UK Census, London Datastore
- Several sources come with information that can be merged across different sources (e.g. geo-codes)