



Data Handling: Import, Cleaning and Visualisation

Lecture 5:

“Big Data” from the Web

Prof. Dr. Ulrich Matter

15/10/2020

Updates

GitHub Classroom back on track

- Due to an update of the system that runs GitHub Classroom, courses around the world experienced problems with autograding in the last week.
- We apologize for any inconveniences that this might have caused!
- For the moment, it looks like we could fix the problem. Please write Philine in case you experience technical issues when committing your solutions in GH Classroom.

Guest lecture: 26.11.2020



Dr. Thomas Spycher (Novalytica AG).

- 45-60 minutes talk.
- Q&A.
- Time for bilateral exchange.

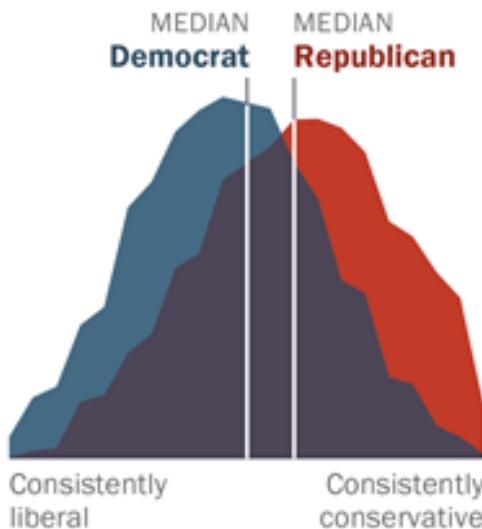
Recap: Research Insights

Background: Increasing political polarization

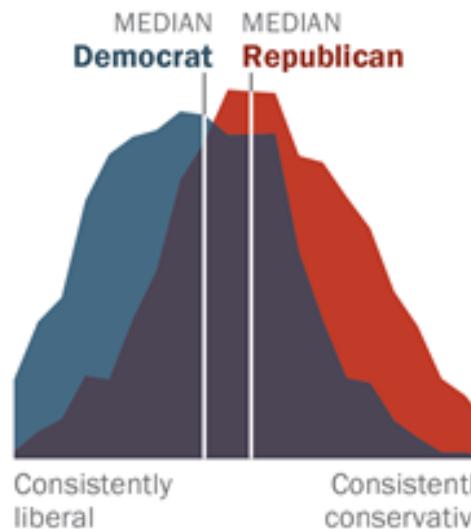
Democrats and Republicans More Ideologically Divided than in the Past

Distribution of Democrats and Republicans on a 10-item scale of political values

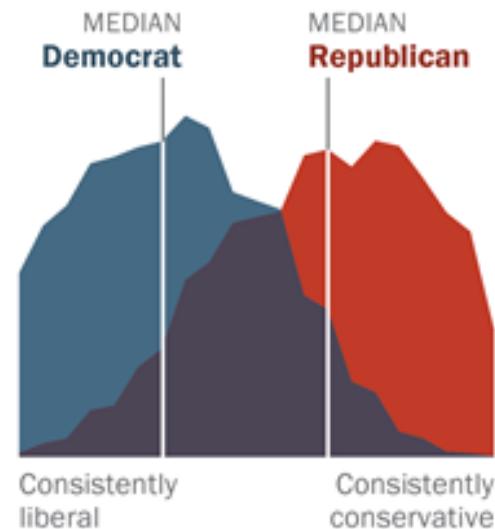
1994



2004



2014



Source: 2014 Political Polarization in the American Public

Notes: Ideological consistency based on a scale of 10 political values questions (see Appendix A). The blue area in this chart represents the ideological distribution of Democrats; the red area of Republicans. The overlap of these two distributions is shaded purple. Republicans include Republican-leaning independents; Democrats include Democratic-leaning independents (see Appendix B).

PEW RESEARCH CENTER

Background: “Algorithmic” polarization in the Web

The screenshot shows a Google search results page for the query "Migration". The search bar at the top contains the word "Migration". Below the search bar, there are navigation links for "All", "Images", "News", "Maps", "Videos", "More", "Settings", and "Tools". A status message indicates "Page 2 of about 668'000'000 results (0.46 seconds)".

The first result is an advertisement from amnesty.ch:

Ad · www.amnesty.ch/
Asyl und Migration: Grundlagen und Begriffe — amnesty.ch
Amnesty unterstützen Jetzt aktiv werden. Spenden Sie jetzt. Über 7 Mio. Unterstützer. Jetzt aktiv werden! Menschenrechte im Fokus. Dienstleistungen: Menschenrechte, Schule & Kurse, Urgent Actions, Spenden Online.
[Spenden](#) - CHF 20.00 - Unterstützen Sie Amnesty · More →

The second result is a link from tdh.ch:

[www.tdh.ch > our-Interventions > migration](#) ▾
Migration | Terre des hommes
Terre des hommes protects children and youth affected by migration, whether it is voluntary or forced. We raise awareness of the related risks while taking into ...

The third result is a link from ilo.org:

[www.ilo.org > global > topics > labour-migration > lang--en](#) ▾
Labour migration (Labour migration) - ILO
According to the ILO global estimates on migrant workers , in 2017, migrant workers accounted for 164 million of the world's approximately 258 million ...

The fourth result is a link from shareweb.ch:

[www.shareweb.ch > site > Migration > focus-area](#) ▾
Current issues on Migration
Jan 21, 2020 - Civil society's key role in the urban governance of migration. 09.10. ... Acting for Awareness: Street Drama Campaign on Safe Labour Migration.

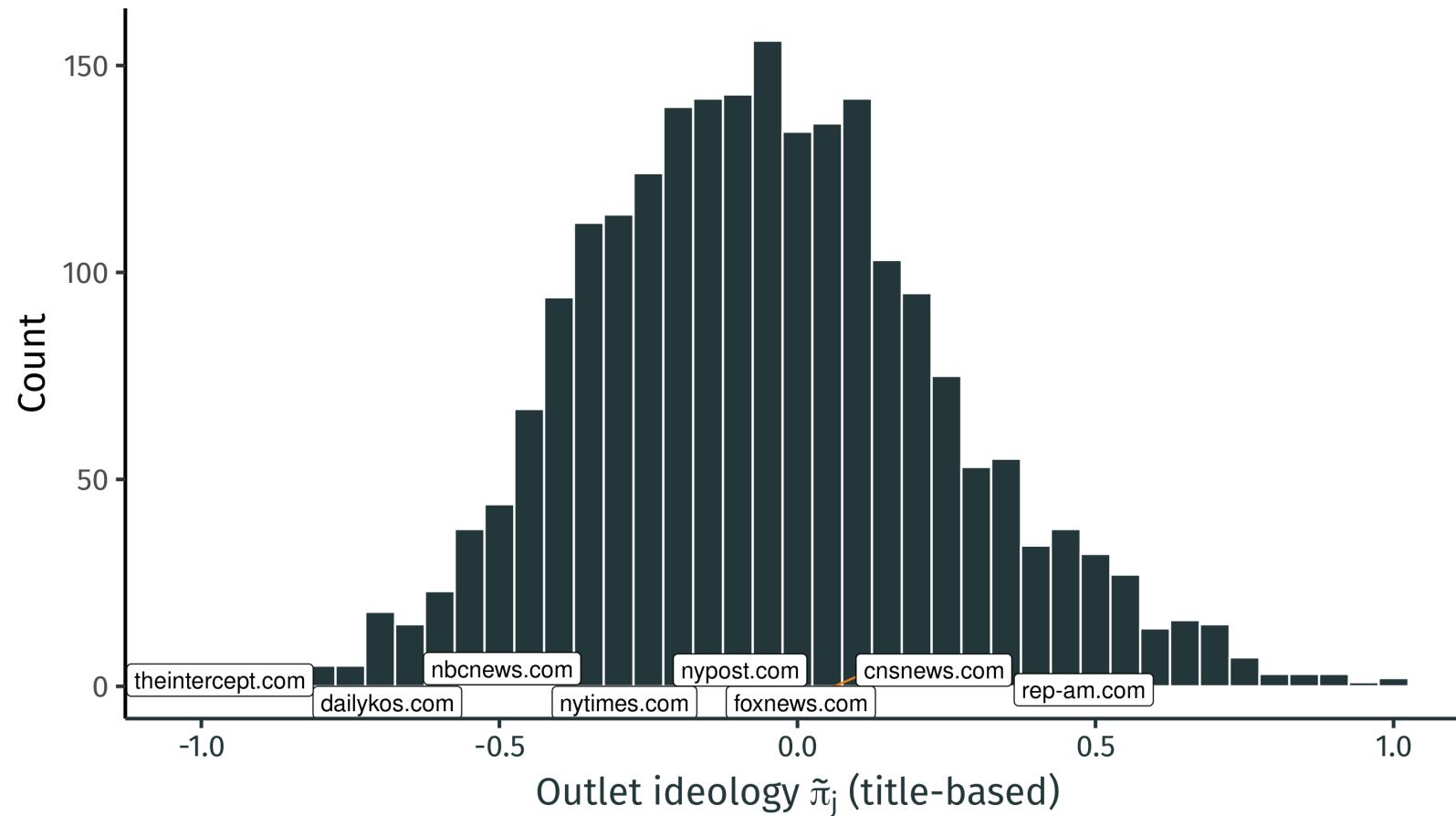
The fifth result is a link from unisg.ch:

[www.unisg.ch > universität > hsgservices > beratung > beratungsstellen](#)
University | Taskforce Migration - University of St.Gallen
To assist the city and the canton with the ongoing flow of refugees, a Taskforce Migration was established at the University of St.Gallen.

The sixth result is a link from letemps.ch:

[www.letemps.ch > theme > migrations](#) ▾ [Translate this page](#)
Tous les articles "Migrations" - Le Temps
frontières Fort afflux de migrants à la frontière grecque après l'ouverture des frontières par ...
Migration Asile: «Les autorités privilient la vitesse à l'équité des ...

Insights: Ideological distribution of US online media sphere



Recap: Computer Code and Data Storage

Computer code

- Instructions to a computer, in a language it understands... (R)
- Code is written to **text files**
- Code is ‘translated’ into 0s and 1s which the CPU can process.

Data storage

- Data is usually stored in **text files**.
 - Read data from text files: data import.
 - Write data to text files: data export.

Inspect a text file

Interpreting 0s and 1s as text...

```
cat helloworld.txt; echo
```

```
## Hello World!
```

Inspect a text file

Directly looking at the 0s and 1s...

```
xxd -b helloworld.txt
```

```
## 00000000: 01001000 01100101 01101100 01101100 01101111 00100000 Hello
## 00000006: 01010111 01101111 01110010 01101100 01100100 00100001 World!
```

Inspect a text file

Directly looking at the 0s and 1s...

```
xxd -b helloworld.txt
```

```
## 00000000: 01001000 01100101 01101100 01101100 01101111 00100000 Hello
## 00000006: 01010111 01101111 01110010 01101100 01100100 00100001 World!
```

How does the computer know that 01001000 is H?

Encoding issues

```
cat hastamanana.txt; echo
```

```
## Hasta Ma?ana!
```

UTF encodings

- 'Universal' standards.
- Contain broad variety of symbols (various languages).
- Less problems with newer data sources...

Take-away message

- **Recognize an encoding issue when it occurs!**
- Problem occurs right at the beginning of the **data pipeline!**
 - Rest of pipeline affected...
 - ... cleaning of data fails ...
 - ... analysis suffers.

Structured Data Formats

- Still text files, but with standardized **structure**.
- **Special characters** define the structure.
- More complex **syntax**, more complex structures can be represented...

CSVs and fixed-width format

- Common format to store and transfer data.
 - Very common in a data analysis context.
- Natural format/structure when data represents a table.

Structures to work with...

- Data structures for storage on hard drive (e.g., csv).
- Representation of data in RAM (e.g. as an R-object)?
 - What is the representation of the 'structure' once the data is parsed (read into RAM)?

Structures to work with (in R)

We distinguish two basic characteristics:

1. Data **types**: integers; real numbers ('numeric values', floating point numbers); text ('string', 'character values').
2. Basic **data structures** in RAM:
 - **Vectors**
 - **Factors**
 - **Arrays/Matrices**
 - **Lists**
 - **Data frames** (very R-specific)

Complex Data Structures

A rectangular data set

father	mother	name	age	gender
		John	33	male
		Julia	32	female
John	Julia	Jack	6	male
John	Julia	Jill	4	female
John	Julia	John jnr	2	male
		David	45	male
		Debbie	42	female
David	Debbie	Donald	16	male
David	Debbie	Dianne	12	female

What is the data about?

A rectangular data set

father	mother	name	age	gender
		John	33	male
		Julia	32	female
John	Julia	Jack	6	male
John	Julia	Jill	4	female
John	Julia	John jnr	2	male
		David	45	male
		Debbie	42	female
David	Debbie	Donald	16	male
David	Debbie	Dianne	12	female

Which observations belong together?

A rectangular data set

father	mother	name	age	gender
		John	33	male
		Julia	32	female
John	Julia	Jack	6	male
John	Julia	Jill	4	female
John	Julia	John jnr	2	male
		David	45	male
		Debbie	42	female
David	Debbie	Donald	16	male
David	Debbie	Dianne	12	female

Can a parser understand which observations belong together?

Limitations of rectangular data

- Only **two dimensions**.
 - Observations (rows)
 - Characteristics/variables (columns)

Limitations of rectangular data

- Only **two dimensions**.
 - Observations (rows)
 - Characteristics/variables (columns)
- Hard to represent hierarchical structures.
 - Might introduce redundancies.
 - Machine-readability suffers (standard parsers won't recognize it).

Alternative formats

- JavaScript Object Notation (JSON)
- Extensible Markup Language (XML)
- and more...

Alternative formats

- JavaScript Object Notation (JSON)
- Extensible Markup Language (XML)
- Origin and most common domain of application: The Web!
 - Need to **transfer** complex data (between machines).
 - Need to **embed** complex data (in human friendly layout).

Deciphering XML

Revisiting COVID-19 data

dateRep,day,month,year,cases,deaths,countriesAndTerritories,geoId,countryterritoryCode
14/10/2020,14,10,2020,66,0,Afghanistan,AF,AFG,38041757,Asia,1.94523087
13/10/2020,13,10,2020,129,3,Afghanistan,AF,AFG,38041757,Asia,1.81116766
12/10/2020,12,10,2020,96,4,Afghanistan,AF,AFG,38041757,Asia,1.50361089

Revisiting COVID-19 data (in XML!)

```
<records>
<record>
<dateRep>14/10/2020</dateRep>
<day>14</day>
<month>10</month>
<year>2020</year>
<cases>66</cases>
<deaths>0</deaths>
<countriesAndTerritories>Afghanistan</countriesAndTerritories>
<geoId>AF</geoId>
<countryterritoryCode>AFG</countryterritoryCode>
<popData2019>38041757</popData2019>
<continentExp>Asia</continentExp>
<Cumulative_number_for_14_days_of_COVID-19_cases_per_100000>1.94523087</Cumulative_ni
</record>
<record>
<dateRep>13/10/2020</dateRep>
...
</records>
```

Revisiting Point Nemo (in XML!)

```
<records>
<record>
<dateRep>14/10/2020</dateRep>
<day>14</day>
<month>10</month>
<year>2020</year>
<cases>66</cases>
<deaths>0</deaths>
<countriesAndTerritories>Afghanistan</countriesAndTerritories>
<geoId>AF</geoId>
<countryterritoryCode>AFG</countryterritoryCode>
<popData2019>38041757</popData2019>
<continentExp>Asia</continentExp>
<Cumulative_number_for_14_days_of_COVID-19_cases_per_100000>1.94523087</Cumulative_ni
</record>
<record>
<dateRep>13/10/2020</dateRep>
...
</records>
```

What features does the format have? What is its logic/syntax?

XML syntax

```
<records>
<record>
<dateRep>14/10/2020</dateRep>
<day>14</day>
<month>10</month>
<year>2020</year>
<cases>66</cases>
<deaths>0</deaths>
<countriesAndTerritories>Afghanistan</countriesAndTerritories>
<geoId>AF</geoId>
<countryterritoryCode>AFG</countryterritoryCode>
<popData2019>38041757</popData2019>
<continentExp>Asia</continentExp>
<Cumulative_number_for_14_days_of_COVID-19_cases_per_100000>1.94523087</Cumulative_ni
</record>
<record>
<dateRep>13/10/2020</dateRep>
...
</records>
```

XML syntax

The actual content we know from the csv-type example above is nested between the 'records'-tags:

```
<records>  
...  
</records>
```

XML syntax: Point Nemo example

Comparing the actual content between these tags with the csv-type format above, we further recognize that there are two principal ways to link variable names to values.

```
<variable>Monthly Surface Clear-sky Temperature (ISCCP) (Celsius)</variable>
<filename>ISCCPMonthly_avg.nc</filename>
<filepath>/usr/local/fer_data/data/</filepath>
<badflag>-1.E+34</badflag>
<subset>48 points (TIME)</subset>
<longitude>123.8W(-123.8)</longitude>
<latitude>48.8S</latitude>
<case date="16-JAN-1994" temperature="9.200012" />
<case date="16-FEB-1994" temperature="10.70001" />
<case date="16-MAR-1994" temperature="7.5" />
<case date="16-APR-1994" temperature="8.100006" />
```

XML syntax

1. Define opening and closing XML-tags with the variable name and surround the value with them, such as in

```
<filename>ISCCPMonthly_avg.nc</filename>.
```

2. Encapsulate the values within one tag by defining tag-attributes such as in <case date="16-JAN-1994" temperature="9.200012" />.

XML syntax

Attributes-based:

```
<case date="16-JAN-1994" temperature="9.200012" />
<case date="16-FEB-1994" temperature="10.70001" />
<case date="16-MAR-1994" temperature="7.5" />
<case date="16-APR-1994" temperature="8.100006" />
```

XML syntax

Tag-based:

```
<cases>
  <case>
    <date>16-JAN-1994</date>
    <temperature>9.200012</temperature>
  </case>
  <case>
    <date>16-FEB-1994</date>
    <temperature>10.70001</temperature>
  </case>
  <case>
    <date>16-MAR-1994</date>
    <temperature>7.5</temperature>
  </case>
  <case>
    <date>16-APR-1994</date>
    <temperature>8.100006</temperature>
  </case>
</cases>
```

Insights: CSV vs. XML

Note the key differences of storing data in XML format in contrast to a flat, table-like format such as CSV:

- Represent much more **complex (multi-dimensional)** data in XML-files than what is possible in CSVs.
 - Arbitrarily complex nesting structure.
 - Flexibility to label tags.

Insights: CSV vs. XML

Note the key differences of storing data in XML format in contrast to a flat, table-like format such as CSV:

- Self-explanatory syntax: **machine-readable and human-readable**.
 - Parsers/computers can more easily handle complex data structures.
 - Humans can intuitively understand what the data is all about just by looking at the raw XML file.

Insights: CSV vs. XML

Potential drawback of XML: **inefficient** storage.

- Tags are part of the syntax, thus, part of the actual file.
 - Tags (variable labels) are **repeated** again and again!
 - CSV: variable labels are mentioned once.
 - Potential solution: data compression (e.g., zip).
- If the data is actually two dimensional, a CSV is more practical.

Deciphering JSON

JSON syntax

- Key difference to XML: no tags, but **attribute-value pairs**.
- A substitute for XML (often encountered in similar usage domains).

XML:

```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumber>
    <type>home</type>
    <number>212 555-1234</number>
  </phoneNumber>
  <phoneNumber>
    <type>fax</type>
    <number>646 555-4567</number>
  </phoneNumber>
  <gender>
    <type>male</type>
  </gender>
</person>
```

XML:

```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</st
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumber>
    <type>home</type>
    <number>212 555-1234</number>
  </phoneNumber>
  <phoneNumber>
    <type>fax</type>
    <number>646 555-4567</number>
  </phoneNumber>
  <gender>
    <type>male</type>
  </gender>
</person>
```

```
{"firstName": "John",
"lastName": "Smith",
"age": 25,
"address": {
  "streetAddress": "21 2nd Street",
  "city": "New York",
  "state": "NY",
  "postalCode": "10021"
},
"phoneNumber": [
  {
    "type": "home",
    "number": "212 555-1234"
  },
  {
    "type": "fax",
    "number": "646 555-4567"
  }
],
"gender": {
  "type": "male"
}}
```

JSON:

XML:

```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
</person>
```

JSON:

```
{"firstName": "John",
  "lastName": "Smith",
}
```

Parsing XML and JSON in R

Parsing XML in R

The following examples are based on the example code shown above (the two text-files `persons.json` and `persons.xml`)

```
# load packages
library(xml2)

# parse XML, represent XML document as R object
xml_doc <- read_xml("persons.xml")
xml_doc

## {xml_document}
## <person>
## [1] <firstName>John</firstName>
## [2] <lastName>Smith</lastName>
## [3] <age>25</age>
## [4] <address>\n  <streetAddress>21 2nd Street</streetAddress>\n  <city>New York</c
## [5] <phoneNumber>\n    <type>home</type>\n    <number>212 555-1234</number>\n</phon
## [6] <phoneNumber>\n    <type>fax</type>\n    <number>646 555-4567</number>\n</phon
## [7] <gender>\n    <type>male</type>\n</gender>
```

Parsing JSON in R

```
# load packages
library(jsonlite)

# parse the JSON-document shown in the example above
json_doc <- fromJSON("persons.json")

# check the structure
str(json_doc)

## Warning: package 'jsonlite' was built under R version 3.6.2

## List of 6
## $ firstName   : chr "John"
## $ lastName    : chr "Smith"
## $ age         : int 25
## $ address     :List of 4
##   ..$ streetAddress: chr "21 2nd Street"
##   ..$ city        : chr "New York"
##   ..$ state       : chr "NY"
##   ..$ postalCode  : chr "10021"
## $ phoneNumber:'data.frame': 2 obs. of  2 variables:
##   ..$ type  : chr [1:2] "home" "fax"
##   ..$ number: chr [1:2] "212 555-1234" "646 555-4567"
## $ gender      :List of 1
##   ..$ type: chr "male"
```

HTML: Computer Code Meets Data

HTML: Code to build webpages

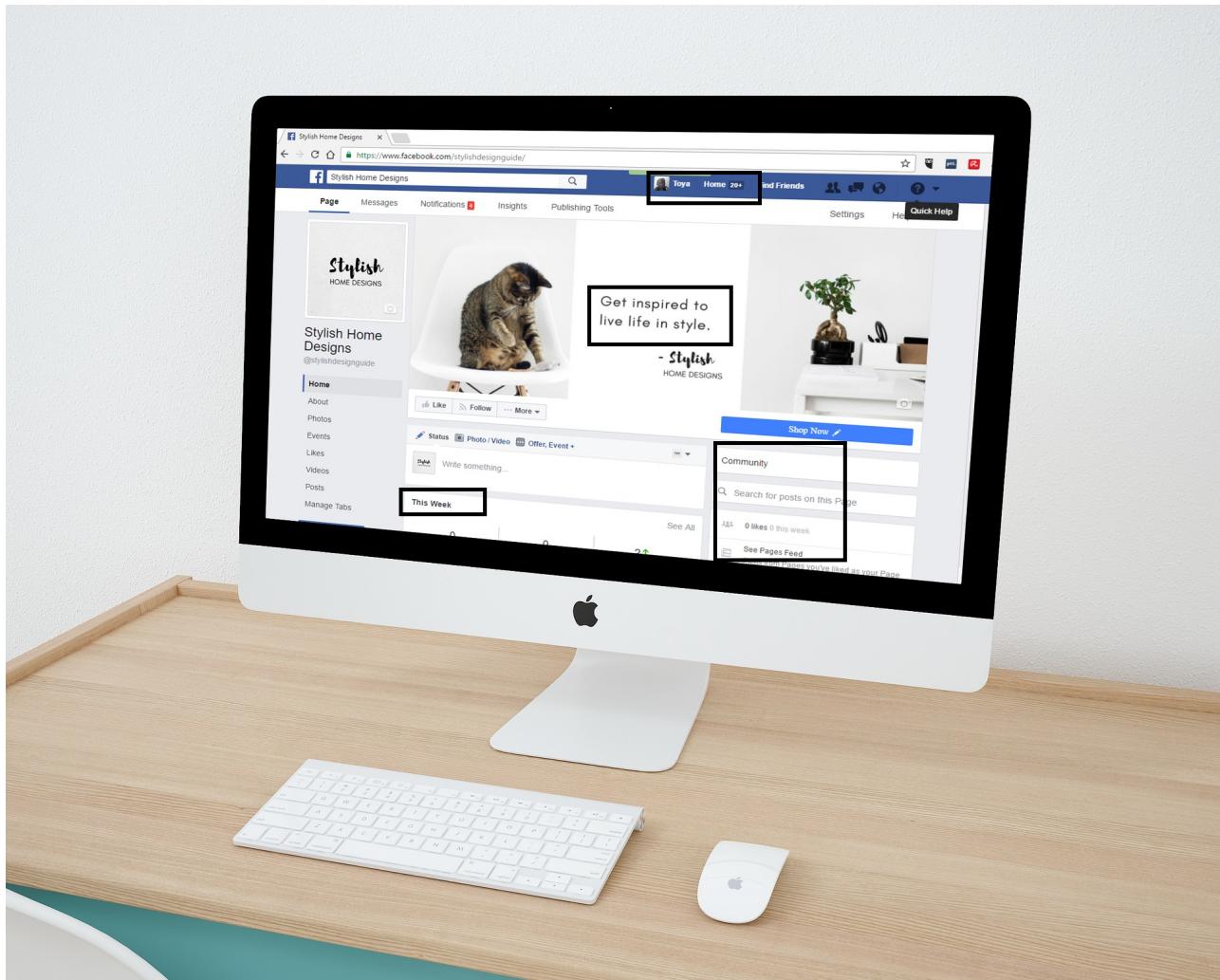
HyperText Markup Language (HTML), designed to be read by a web browser.



HTML: Code to build webpages



HTML documents contain data!



HTML documents: code and data!

- Web designer's perspective: "HTML is a tool to design the layout of a webpage (and the resulting HTML document is **code**)."
- From a data scientist's perspective: "HTML gives the **data** contained in a webpage (the actual content) a certain degree of structure which can be exploited to systematically extract the data from the webpage."

HTML documents: code and data!

HTML documents/webpages consist of '**semi-structured data**':

- A webpage can contain a HTML-table (**structured data**)...
- ...but likely also contains just raw text (**unstructured data**).

```
<!DOCTYPE html>

<html>
  <head>
    <title>hello, world</title>
  </head>
  <body>
    <h2> hello, world </h2>
  </body>
</html>
```

Similarities to other formats?

Characteristics of HTML

1. Annotate/'mark up' data/text (with tags)

- Defines **structure** and hierarchy
- Defines content (pictures, media)

Characteristics of HTML

1. **Annotate/'mark up'** data/text (with tags)

- Defines **structure** and hierarchy
- Defines content (pictures, media)

2. **Nesting** principle

- `head` and `body` are nested within the `html` document
- Within the `head`, we define the `title`, etc.

Characteristics of HTML

1. **Annotate/'mark up'** data/text (with tags)

- Defines **structure** and hierarchy
- Defines content (pictures, media)

2. **Nesting** principle

- `head` and `body` are nested within the `html` document
- Within the `head`, we define the `title`, etc.

3. Expresses what is what in a document.

- Doesn't explicitly 'tell' the computer what to do
- HTML is a markup language, not a programming language.

HTML document as a ‘tree’

- ‘Root’: <html>...</html>
- ‘Children’ of the root node: <head>...</head>, <body>...</body>
- ‘Siblings’ of each other: <head>...</head>, <body>...</body>

HTML document as a ‘tree’

HTML (DOM) tree diagram (by Lubaochuan 2014, licensed under the [Creative Commons Attribution-Share Alike 4.0 International license](#)).

Two ways to read a webpage into R

In this example, we look at [Wikipedia's Economy of Switzerland page.](#)

Read the HTML line-by-line

```
swiss_econ <- readLines("https://en.wikipedia.org/wiki/Economy_of_Switzerland")

## Warning in readLines("https://en.wikipedia.org/wiki/Economy_of_Switzerland"): inco
## found on 'https://en.wikipedia.org/wiki/Economy_of_Switzerland'

head(swiss_econ)

## [1] "<!DOCTYPE html>"
## [2] "<html class=\"client-nojs\" lang=\"en\" dir=\"ltr\">"
## [3] "<head>"
## [4] "<meta charset=\"UTF-8\"/>"
## [5] "<title>Economy of Switzerland – Wikipedia</title>"
## [6] "<script>document.documentElement.className=\"client-js\";RLCONF={\"wgBreakFr...
```

Navigate the raw HTML

Search for specific content

```
line_number <- grep('US Dollar Exchange', swiss_econ)
```

```
line_number
```

```
## [1] 216
```

```
swiss_econ[line_number]
```

```
## [1] "<th>US Dollar Exchange"
```

What are we missing here?

Parse the HTML!

- Navigate the document like an XML document!
 - Same logic (but tags are pre-defined)...
 - Traverse the tree.
- Access specific parts of the contained data directly.

Make use of the structure!

Parsing a Webpage with R

```
# install package if not yet installed
# install.packages("rvest")

# load the package
library(rvest)

# parse the webpage, show the content
swiss_econ_parsed <- read_html("https://en.wikipedia.org/wiki/Economy_of_Switzerland")
swiss_econ_parsed

## {html_document}
## <html class="client-nojs" lang="en" dir="ltr">
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">\n<
## [2] <body class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject mw-ed
```

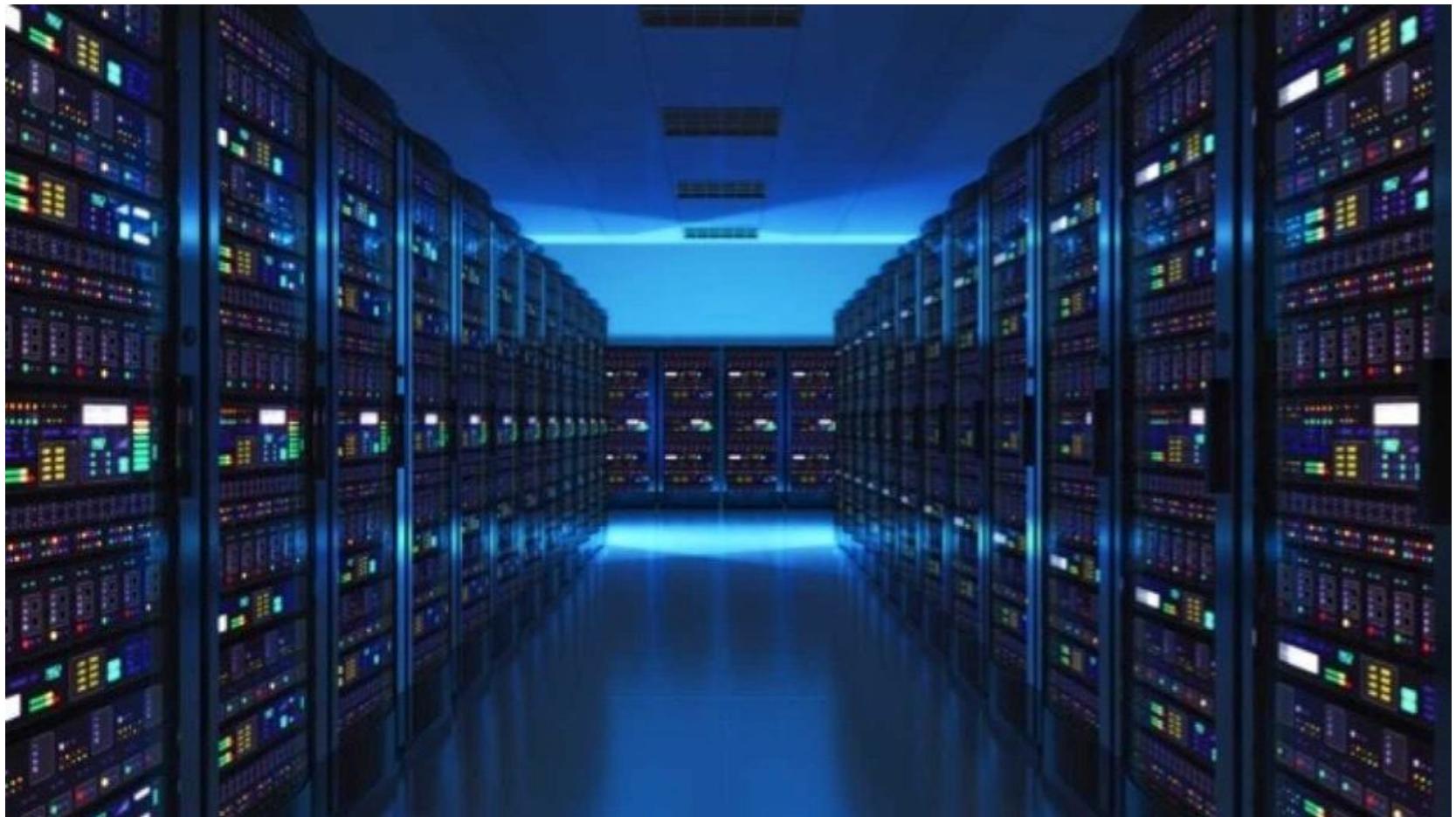
Parsing a Webpage with R

Now we can easily separate the data/text from the html code. For example, we can extract the HTML table containing the data we are interested in as a `data.frames`.

```
tab_node <- html_node(swiss_econ_parsed,  
                      xpath = "//*[@id='mw-content-text']/div/table[2]")  
tab <- html_table(tab_node)  
tab
```

```
##      Year GDP (billions of CHF) US Dollar Exchange  
## 1    1980          184     1.67 Francs  
## 2    1985          244     2.43 Francs  
## 3    1990          331     1.38 Francs  
## 4    1995          374     1.18 Francs  
## 5    2000          422     1.68 Francs  
## 6    2005          464     1.24 Francs  
## 7    2006          491     1.25 Francs  
## 8    2007          521     1.20 Francs  
## 9    2008          547     1.08 Francs  
## 10   2009          535     1.09 Francs  
## 11   2010          546     1.04 Francs  
## 12   2011          659     0.89 Francs  
## 13   2012          632     0.94 Francs  
## 14   2013          635     0.93 Francs  
## 15   2014          644     0.92 Francs  
## 16   2015          646     0.96 Francs
```

Why should we care?



Source: <https://www.sciencefocus.com/future-technology/how-much-data-is-on-the-internet/>.

How much data is stored on the Web?

“One way to answer this question is to consider the sum total of data held by all the big online storage and service companies like Google, Amazon, Microsoft and Facebook. Estimates are that **the big four store at least 1,200 petabytes** between them. That is **1.2 million terabytes (one terabyte is 1,000 gigabytes)**.” (Gareth Mitchell, ScienceFocus)

Q&A

References