Data Handling: Import, Cleaning and Visualisation

Lecture 5:

Webdata, Text, and Images

Prof. Dr. Ulrich Matter

Structured Data Formats

Still text files, but with standardized **structure**.

Special characters define the structure.

More complex **syntax**, more complex structures can be represented…

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)

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?

Revisiting COVID-19 data

```
dateRep,day,month,year,cases,deaths,countriesAndTerritories,geoId,countryterritoryCo
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_nu
</record>
<record>
<dateRep>13/10/2020</dateRep>

...
</records>
```

Revisiting COVID-19 (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_nu
</record>
<record>
<dateRep>13/10/2020</dateRep>

...
</records>
```

What features does the format have? What is it's 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_nu
</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: Temperature Data example

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

Represent much more **complex (multi-dimensional)** data in XML-files than what is possible in CSVs.

Self-explanatory syntax: **machine-readable and human-readable**.

Tags are part of the syntax, give both structure and name variables.

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:

```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>
```

JSON:

```json
{"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"
  }
}
```

XML:

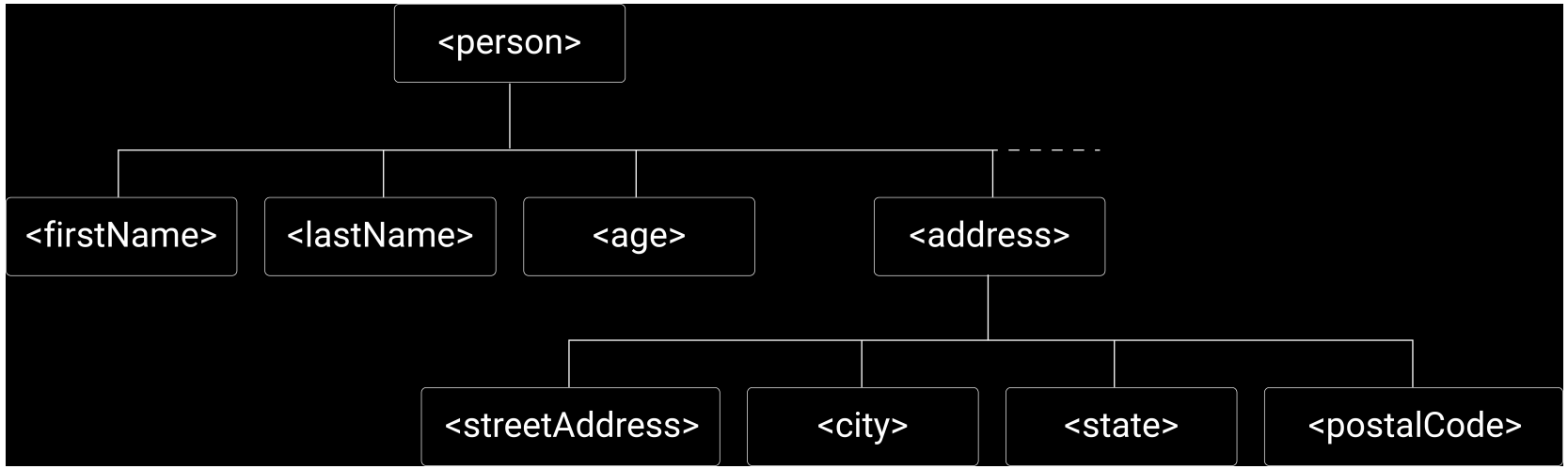
```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>

</person>
```

JSON:

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

}
```


<person>
<firstName>
<lastName>
<age>
<address>
<streetAddress>
<city>
<state>
<postalCode>

HTML: Code to build webpages

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

HTML documents contain data!

HTML documents: code and data!

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

A webpage can contain a HTML-table (**structured data**)…

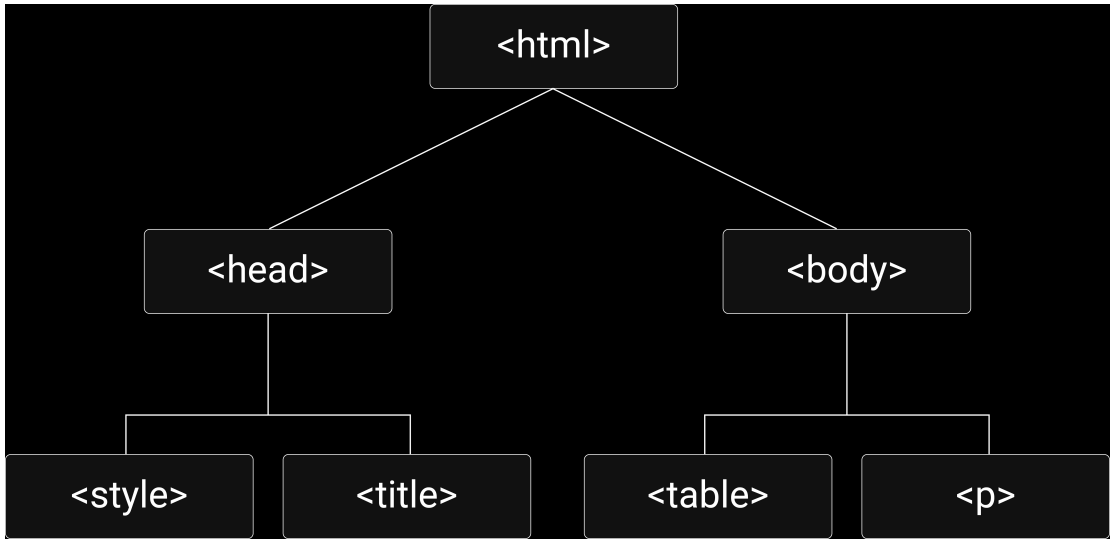
…but likely also contains just raw text (**unstructured data**).

```
<!DOCTYPE html>

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

Similarities to other formats?

HTML document as a ‘tree’

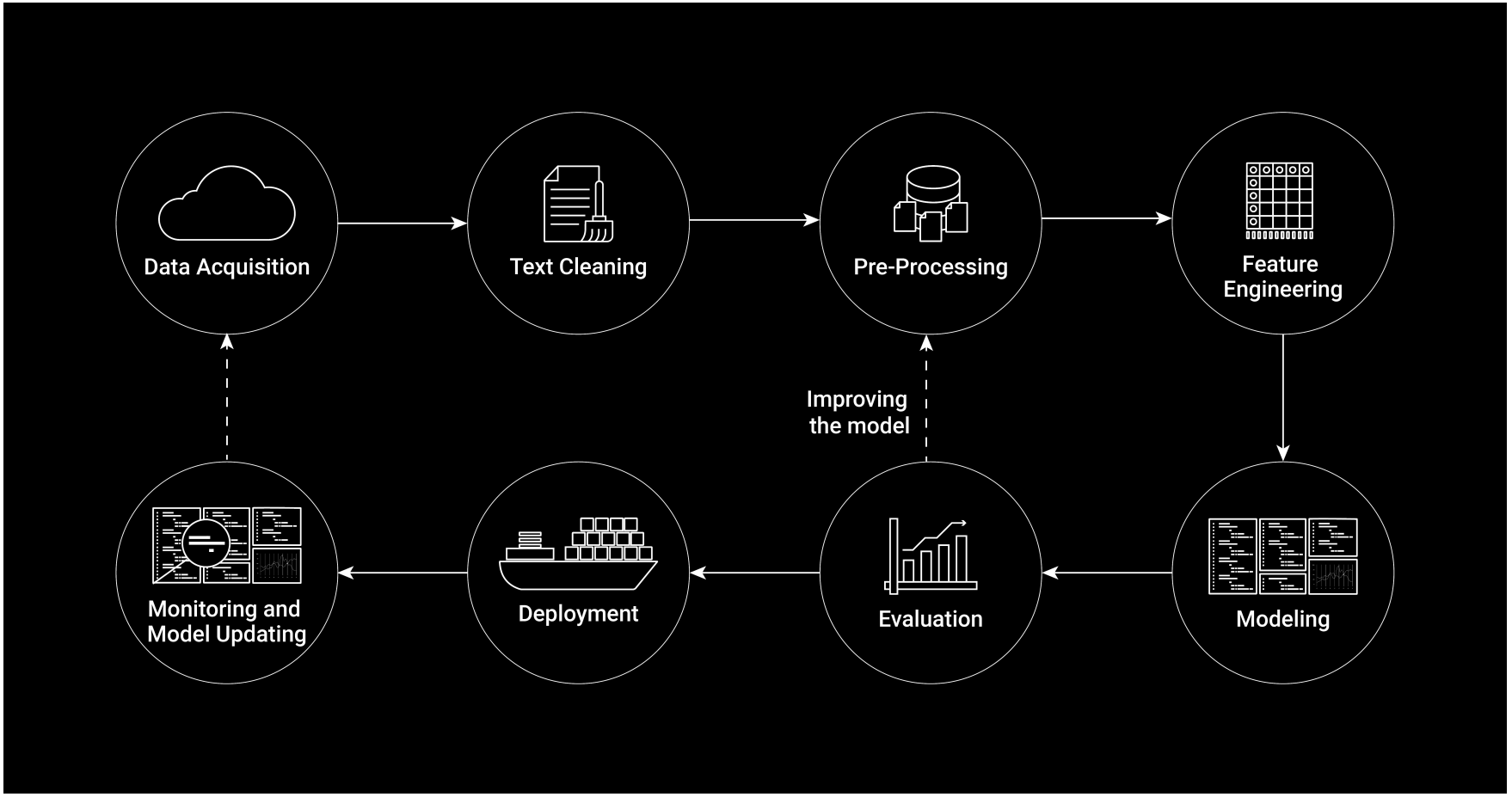


Two ways to read a webpage into R

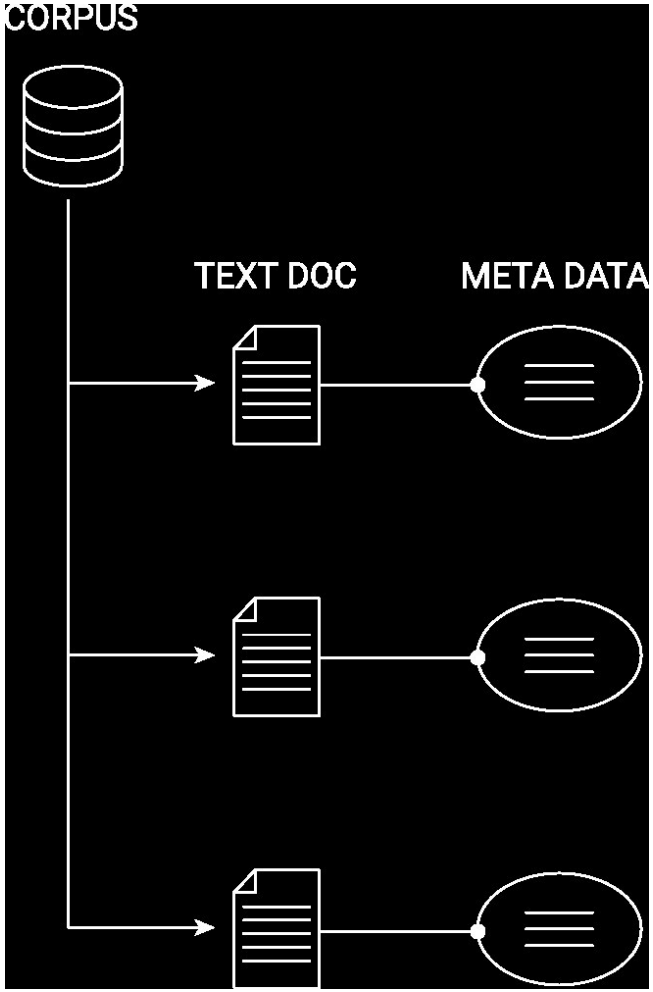
In this example, we look at Wikipedia's Economy of Switzerland page.

Year	GDP (billions of CHF)	US Dollar Exchange
1980	184	1.67 Francs
1985	244	2.43 Francs
1990	331	1.38 Francs
1995	374	1.18 Francs
2000	422	1.68 Francs
2005	464	1.24 Francs
2006	491	1.25 Francs
2007	521	1.20 Francs
2008	547	1.08 Francs
2009	535	1.09 Francs
2010	546	1.04 Francs
2011	659	0.89 Francs
2012	632	0.94 Francs
2013	635	0.93 Francs
2014	644	0.92 Francs
2015	646	0.96 Francs
2016	659	0.98 Francs
2017	668	1.01 Francs

Handling text data for analysis



Data structure: text corpus



Working with text data in R: Quanteda

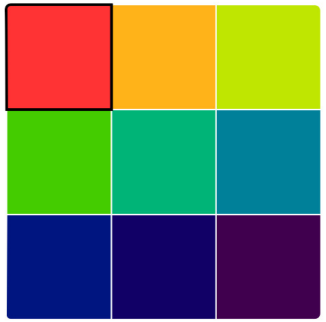
Basic data structures

Raster images: a matrix of pixels, as well as the color of each pixel.

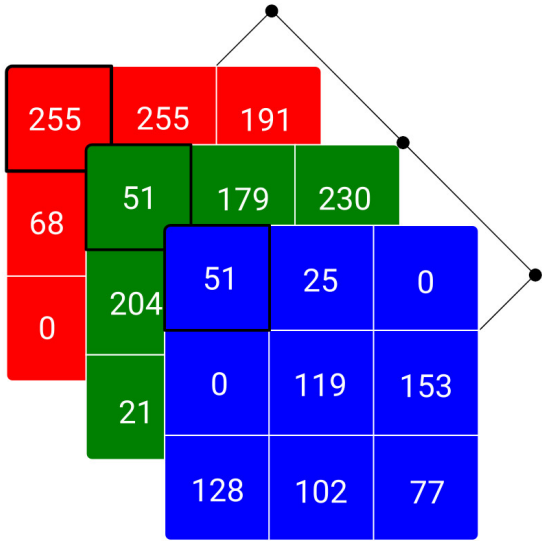
Vector-based images: text files that store the coordinates of points on a surface and how these dots are connected (or not) by lines.

Raster images

A



B



Raster and vector images in R

Use cases in economic research and beyond

Extract text from historical documents (scan, use OCR)

Use machine learning to label text (too costly to do manually)

Extract information from maps

“Donald Trump riding a donkey while eating pizza”

“Donald Trump riding a donkey while eating pizza”

“Donald Trump riding a donkey while eating pizza”

“Donald Trump riding a donkey while eating pizza”

“Data Cleaning”

“Data Cleaning”

“A large pile of data”

“A person analyzing economic data”

References