

Methods for unstructured data

Lecture 2: Web scraping

Helge Liebert

Roadmap and goals

- Use processing of web data to introduce various ideas along the way.
- Accessing data on the web: APIs.
- Gathering (semi-structured) web data and transforming it into structured data (*“web scraping”*).
- Give you a general understanding how web scrapers work.
- Foster an understanding of the development process.
- Point you towards the necessary tools so you can write your own.

1. Introduction
2. Static and dynamic websites
3. Application Programming Interfaces
4. The Document Object Model
5. Examples
6. Assignment

Guiding problem


- How to turn unstructured into structured data?
- Consider a situation where
 - You want to get data from the internet.
 - The data is in unstructured/semi-structured form.
 - Possibly embedded in a website.
 - You want to transform it into a differently structured format for further use.
 - You need to filter the available information.

[illegible]


...and this

NSCB - Good Governance x Our Campaigns - Candidates x

www.nscb.gov.ph/ggi/details_prov.asp



REPUBLIC OF THE PHILIPPINES
PHILIPPINE STATISTICS AUTHORITY
SOLID • RESPONSIVE • WORLD-CLASS


Good Governance Index

GGI Indicators for the province of Misamis Oriental

Indicator	2005		2008	
	Value	Rank 1a	Value	Rank 1a
Economic Governance Index	107.08	37	86.12	71
Total Financial Resources Generated per Capita (in Millions)	171.05	45	233.43	50
Per Capita Tax and Non-Tax (in Millions)	71.66	1	73.87	7
Per Capita Total Deposits	122.61	4	N/A	N/A
Per Capita Expenditure on Social Services	212.80	22	90.22	67
Unemployment Rate	125.56	47	125.56	47
Underemployment Rate	55.31	53	55.31	53
Inflation Rate	66.37	14	37.46	37
Poverty Gap	87.56	34	72.64	42
Poverty Incidence	84.90	27	88.10	31
Political Governance Index	103.24	13	99.86	18
Crime Solution Efficiency Rate	103.91	8	96.76	23
Voter's Turnout Rate	102.56	31	102.56	31
Administrative Governance Index	128.00	23	154.38	13
A. Education Index	80.79	68	107.25	52
Elementary Teacher to Pupil Ratio	100.07	55	96.93	54
High School Teacher to Pupil Ratio	100.93	45	99.86	53
No. of Public Elem. Schools per 1000 School Age Population	56.02	85	96.64	45
No. of Public High Schools per 1000 School Age Population	51.16	70	118.54	60
Enrollment in Gov't. Elem. Sch. per 1000 School Age Population	57.08	75	95.78	9
Enrollment in Gov't. HS. Sch. per 1000 School Age Population	61.13	63	140.90	8
Cohort Survival Rate (Elementary)	93.57	46	83.99	49
Cohort Survival Rate (High School)	79.85	32	85.29	48
Elementary Pupil - Classroom Ratio	102.75	58	103.88	54
High School Pupil - Classroom Ratio	105.37	61	140.74	53
B. Health Index	229.66	19	212.88	21
Total Health Personnel per 1000 Population	193.11	19	170.55	23
% Birth less the 2500 grams	478.47	12	393.89	21
% of Household with access to safe water	109.09	48	131.13	1
% Barangay Health Station per 100,000 Population	137.99	30	151.93	21
C. Power and ICT Index	76.26	35	144.00	1
Power Index	124.83	1	124.83	1
ICT Index (Telephone Density per 1000 Population)	27.89	41	157.44	1
Good Governance Index	116.27	33	117.31	42

1a 1=highest/best, out of 79 provinces

Good Governance Index
[Main Page](#)
[Database](#)
[Technical Notes](#)
[Press Release](#)

...to this.

master.csv - LibreOffice Calc																			
File Edit View Insert Format Tools Data Window Help																			
Liberation Sans 10																			
A1																			
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
id	name	jahr	land	provinz	gruppe	popcat	pop	ehe	lebgeb	lebgebo	lebgebot	todgebo	todgebot	geb	gebo	todgeb	todgebshare	ix	
1	Aachen	1927	Preussen	Rheinprovinz	A	pop >= 100,000	156360	1423	2753	2366	387	80	7	2840	2446	87	3.0633902		
2	Aachen	1928	Preussen	Rheinprovinz	A	pop >= 100,000	150991	1446	2723	2260	463	66	17	2806	2326	83	2.9579473		
3	Aachen	1929	Preussen	Rheinprovinz	A	pop >= 100,000	155542	1558	2635	2192	443	49	23	2707	2241	72	2.659771		
4	Aachen	1930	Preussen	Rheinprovinz	A	pop >= 100,000	154634	1501	2635	2189	446	46	22	2703	2235	68	2.5157232		
5	Aachen	1931	Preussen	Rheinprovinz	A	pop >= 100,000	154400	1408	2429	2030	399	53	22	2504	2083	75	2.9952078		
6	Aachen	1932	Preussen	Rheinprovinz	A	pop >= 100,000	153834	1426	2305	1968	337	46	20	2371	2014	66	2.7836356		
7	Aachen	1933	Preussen	Rheinprovinz	A	pop >= 100,000	162990	1616	2371	2028	343	54	23	2448	2082	77	3.1454248		
8	Aachen	1934	Preussen	Rheinprovinz	A	pop >= 100,000	163839	1942	2841	2451	480	76	8	3025	2527	84	2.7768595		
9	Aachen	1935	Preussen	Rheinprovinz	A	pop >= 100,000	164180	1570	3048	2516	532	54	21	3123	2570	75	2.4015369		
10	Aachen	1936	Preussen	Rheinprovinz	A	pop >= 100,000	164105	1502	3012	2382	630	43	25	3080	2425	68	2.2077923		
12	Ahlen	1927	Preussen	Westfalen	D	15,000 <= pop < 30,000	23956	239	606	596	10	27	1	634	623	28	4.4164038		
13	Ahlen	1928	Preussen	Westfalen	D	15,000 <= pop < 30,000	24703	266	625	621	4	18	0	643	639	18	2.7993779		
14	Ahlen	1929	Preussen	Westfalen	D	15,000 <= pop < 30,000	25043	227	624	609	15	26	0	650	635	26	4		
15	Ahlen	1930	Preussen	Westfalen	D	15,000 <= pop < 30,000	25226	202	568	548	20	25	1	594	573	26	4.3771043		
16	Ahlen	1931	Preussen	Westfalen	D	15,000 <= pop < 30,000	25373	191	508	483	25	34	0	542	517	34	6.2730627		
17	Ahlen	1932	Preussen	Westfalen	D	15,000 <= pop < 30,000	25549	200	550	514	36	13	1	564	527	14	2.4822695		
18	Ahlen	1933	Preussen	Westfalen	D	15,000 <= pop < 30,000	25153	301	478	437	41	15	2	495	452	17	3.4343433		
19	Ahlen	1934	Preussen	Westfalen	D	15,000 <= pop < 30,000	25700	283	658	606	52	19	2	679	625	21	3.0927835		
20	Ahlen	1935	Preussen	Westfalen	D	15,000 <= pop < 30,000	25937	212	661	621	40	17	0	678	638	17	2.5073745		
21	Ahlen	1936	Preussen	Westfalen	D	15,000 <= pop < 30,000	26104	228	673	625	48	11	1	685	636	12	1.7518249		
22	Allenstein	1927	Preussen	Ostpreussen	C	30,000 <= pop < 50,000	39315	241	892	859	33	14	3	909	873	17	1.870187		
23	Allenstein	1928	Preussen	Ostpreussen	C	30,000 <= pop < 50,000	39232	293	943	898	45	24	2	969	922	26	2.6831784		
24	Allenstein	1929	Preussen	Ostpreussen	C	30,000 <= pop < 50,000	39114	267	959	900	56	21	3	924	855	25	2.5406504		
25	Allenstein	1930	Preussen	Ostpreussen	C	30,000 <= pop < 50,000	39345	283	862	810	52	14	5	881	824	19	2.1566403		
26	Allenstein	1931	Preussen	Ostpreussen	C	30,000 <= pop < 50,000	39876	301	884	836	48	17	4	905	853	21	2.320442		
27	Allenstein	1932	Preussen	Ostpreussen	C	30,000 <= pop < 50,000	40078	323	837	796	41	15	1	853	811	16	1.8757327		
28	Allenstein	1933	Preussen	Ostpreussen	C	30,000 <= pop < 50,000	43079	340	890	854	36	14	4	908	868	18	1.9823788		
29	Allenstein	1934	Preussen	Ostpreussen	C	30,000 <= pop < 50,000	43506	405	1062	1015	47	28	4	1094	1043	32	2.9250457		
30	Allenstein	1935	Preussen	Ostpreussen	C	30,000 <= pop < 50,000	43881	363	1118	1051	67	16	4	1138	1067	20	1.7574693		
31	Allenstein	1936	Preussen	Ostpreussen	C	30,000 <= pop < 50,000	44566	364	1150	1037	82	21	5	1145	1058	26	2.2707424		
32	Altena	1927	Preussen	Westfalen	D	15,000 <= pop < 30,000	15931	131	240	226	14	5	0	245	231	5	2.0408163		
33	Altena	1928	Preussen	Westfalen	D	15,000 <= pop < 30,000	16333	163	244	229	15	11	2	257	240	13	5.0583658		
34	Altena	1929	Preussen	Westfalen	D	15,000 <= pop < 30,000	16464	139	257	248	9	6	1	264	254	7	2.6515152		
35	Altena	1930	Preussen	Westfalen	D	15,000 <= pop < 30,000	16498	139	242	220	22	16	1	259	236	17	5.6537064		
36	Altena	1931	Preussen	Westfalen	D	15,000 <= pop < 30,000	16407	138	196	186	10	5	2	203	191	7	3.4482758		
37	Altena	1932	Preussen	Westfalen	D	15,000 <= pop < 30,000	16115	122	192	176	16	9	0	201	185	9	4.477612		
38	Altena	1933	Preussen	Westfalen	D	15,000 <= pop < 30,000	16133	138	189	174	15	6	1	196	180	7	3.5712695		
39	Altena	1934	Preussen	Westfalen	D	15,000 <= pop < 30,000	16246	162	258	223	35	8	2	268	231	10	3.7313433		

Find

Find All Match Case

Sheet 1 of 1

Default

Sum=0

110%

Goal: Automation

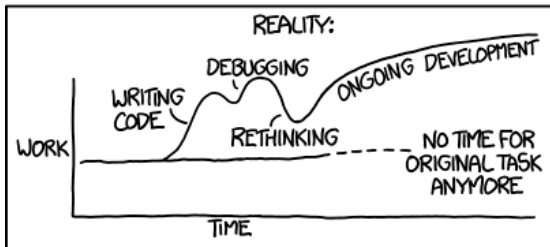
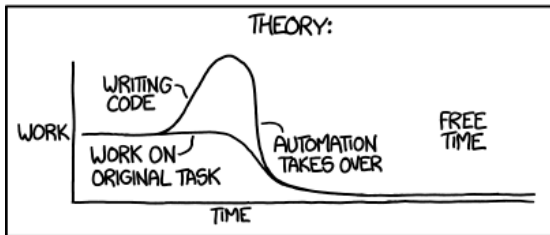
- Digitalization offers exciting data for research. But: *Data is messy*.
- Gathering or processing data often involves repetitive manual tasks.
- Disadvantages:
 - Manual tasks are often not well documented or reproducible ex post.
 - Manual work is frustrating and a huge time-sink.
 - Manual work may not be feasible with large data.

➡ Automation helps!

- Frees you to engage in other work.
- You learn new things.
- Should you encounter the same class of problem in the future, you already have a solution at hand.

Automation

"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"



Getting started—things to consider before you begin

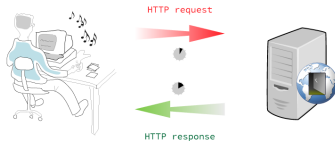
- Pick up the phone and try to get the data directly.
- Search if somebody has already faced the same or a similar problem.
- Does the site or service provide an API that you can access directly?
- Is there a wrapper for it?
- Is the website only online for a limited time? Do you want an original snapshot as a backup? Is it more convenient to filter your data offline?

Static and dynamic websites

Static vs. dynamic websites

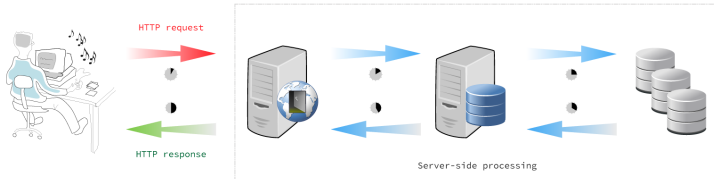
Scheme A

Static Website



Scheme B

Dynamic Website



Save an offline copy

- Use the shell utilities `wget` or `curl` to download the complete site.
- Also useful if you just want a set of files (e.g. pdf documents) from the same site directory.
- Convenient for static sites of limited size.
- Infeasible for large sites or sites that create content dynamically.

Examples

- Simple http GET request.
`wget http://www.google.com`
- Recursively download a website.
`wget -r http://www.some-site.com/some-subdir/`
- Download all pdfs from a site.
`wget -r -A.pdf http://url-to-webpage-with-pdfs/`
- Mirror a site offline and convert links for local browsing.
`wget --mirror -p --convert-links -p ./local-dir
http://target-website.com`

Application Programming Interfaces

- Data providers often offer Web APIs (*Application Programming Interface*) to access data.
- Allow programmable access to data via a defined set of HTTP messages. Similar to visiting a website: you specify a URL and information is sent to your machine.
- With a website, you receive code interpreted by your browser (HTML, CSS, JavaScript). With an API, you receive data.
- Usually in JSON (*JavaScript Object Notation*) or XML (*Extensible Markup Language*) format.

Web APIs

- Often just two steps:
 1. Construct the URL query that serves as the API request.
 2. Process the response message the API sends back.
- Examples:
 - <https://api.kivaws.org/v1/loans/newest.html>
 - <https://api.kivaws.org/v1/loans/newest.json>
 - <https://api.kivaws.org/v1/loans/search.json?sector=Agriculture&country=VN>
 - <https://www.theyworkforyou.com/api/getMPs?&key=someapikeyhere&output=js>
- Libraries may offer wrappers for APIs: **WDI**, **wbstats**, **twfy**, **pvsR**, Google Maps, OpenStreetMap/OSRM, ...
- Sometimes it is possible to reverse engineer a site's internal API rather than scraping the HTML.

The Document Object Model

HTML and the Document Object Model

- Extracting information from the web requires a basic understanding of HTML and the associated Document Object Model (DOM).
- HTML elements provide the structure and content of web pages.
- Consist of `<start>` and `</end>` tags, with content in between.
`<tagname>Content here</tagname>`
- A page consists of nested elements.
- The `html` element is the outer-most element, nesting the `head` and `body` elements, which in turn have nested elements.
- Nesting structure of elements can be represented by a tree (DOM).

Document Object Model

- The DOM is a programming interface for HTML and XML documents.
- Provides a structured representation of the document.
- A document as a group of nodes, each node representing a part of the document.
- Allows programmatic access to the tree to change the structure, style and content of the document.
- Connects web pages to scripts or programming languages.

A simple HTML page

A simple HTML page:

```
<html>
  <head>
    <title>My Web Page</title>
  </head>
  <body>
    <h1>Welcome To My Web Page</h1>
  </body>
</html>
```

How a browser renders this page:

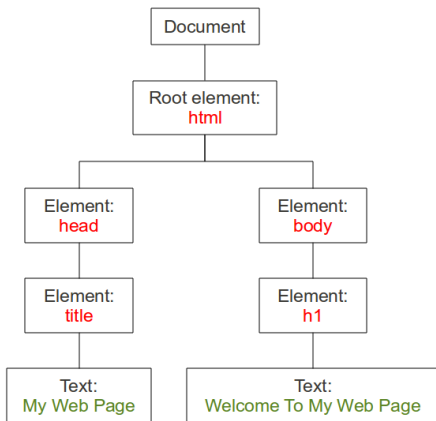


HTML and the DOM

A simple HTML page:

```
<html>
  <head>
    <title>My Web Page
  </title>
</head>
<body>
  <h1>Welcome To My Web Page
</h1>
</body>
</html>
```

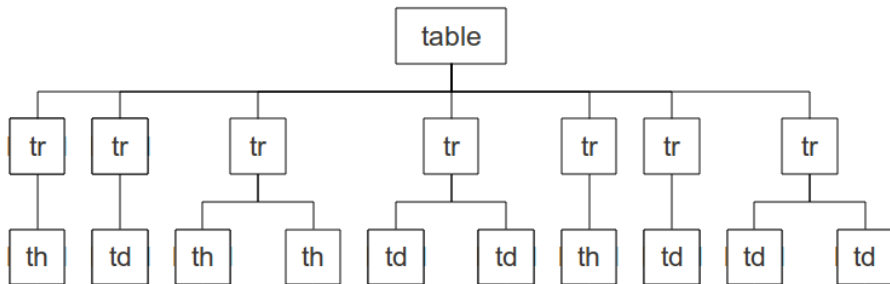
Corresponding node tree:



DOM node trees

- HTML DOM views a document as a tree structure called node tree.
- Everything in an HTML document is a node.
 - The entire document is a document node
 - Every HTML element is an element node
 - Every HTML attribute is an attribute node
 - Text content in the HTML elements is a text node
- Nodes can be accessed through the tree.
- Nodes may be assigned unique id attributes.

Example: An HTML table element



- Tables are represented by a top-level **table** element.
- The **table** element nests **tr** (*table row*) elements.
- These nest **th** (*table header*) and **td** (*table data*) element cells.

HTML and the DOM

- HTML tags can have attributes and text content.

```
<tag attribute="value" attribute2="value">Text content.</tag>
```

- Example page:

```
<html>
  <head>
    <title>My Web Page</title>
  </head>
  <body>
    <h1>Welcome To My Web Page</h1>
    
    <a href="pagelink.html" id="pageid">Check this other page.</a>
  </body>
</html>
```

Data from the web

Infant mortality - Wikipedia

https://en.wikipedia.org/wiki/Infant_mortality

Government and bureaucracies tend to show an insensitivity to these parents and their recent suffering from a lost child, and produce broad disclaimers in the IMR reports that the information has not been properly reported, resulting in these discrepancies. Little has been done to address the underlying structural problems of the vital registry systems in respect to the lack of reporting from parents in rural areas, and it has been created a gap between the official and popular meanings of child death.^[42] It is also argued that the bureaucratic separation of vital death recording from cultural death rituals is to blame for the inaccuracy of the infant mortality rate (IMR). Vital death registries often fail to recognize the cultural implications and importance of infant deaths. It is not to be said that vital registry systems are not an accurate representation of a region's socio-economic situation, but this is only the case if these statistics are valid, which is unfortunately not always the circumstance. "Popular death registries" is an alternative method for collecting and processing statistics on infant and child mortality. Many regions may benefit from "popular death registries" who are culturally linked to infants may be able to provide more accurate statistics on the incidence of infant mortality.^[43] According to ethnographic data, "popular death registries" refers to people who had inside knowledge of ayahuasca, including the grow-logger, galekeeper, midwife, popular healers etc. — all key participants in mortality rituals.^[44] By combining the methods of household surveys, vital registries, and asking "popular death registries" this can increase the validity of child mortality rates, but there are many barriers that can reflect the validity of our statistics of infant mortality. One of these barriers are political economic decisions. Numbers are exaggerated when international funds are being sold and underestimated during reelections.^[45]

The bureaucratic separation of vital death reporting and cultural death rituals stems in part due to structural violence.^[46] Individuals living in rural areas of Brazil need to invest large capital for lodging and travel in order to report infant death to a Brazilian Assistance League office. The negative financial aspects deter registration, as often individuals are of lower income and cannot afford such expenses.^[46] Similar to the lack of birth reporting, families in rural Brazil face difficult choices based on already existing structural arrangements when coming to report infant mortality. Financial constraints such as reliance on food supplementations may also lead to skewed infant mortality data.^[46]

In developing countries such as Brazil the deaths of impoverished infants are regularly unrecorded into the countries vital registration system; this causes a skew statistically. Culturally validity and contextual soundness can be used to ground the meaning of mortality from a statistical standpoint. In northeast Brazil they have accomplished this standpoint while conducting an ethnographic study combined with an alternative method to survey infant mortality.^[47] These types of techniques can develop quality ethnographic data that will ultimately lead to a better portrayal of the magnitude of infant mortality in the region. Political economic reasons have been seen to skew the infant mortality data in the past when governor Caramia denied his presidency campaign by inflating the infant mortality rate during his term in office. By using this new type of surveying, these instances can be minimized and removed, overall creating accurate and sound data.^[47]

Epidemiology

See also: *List of countries by infant mortality rate*

For the world, and for both less developed countries (LDCs) and more developed countries (MDCs), IMR declined significantly between 1960 and 2001. According to the *State of the World's Mothers* report by *Save the Children*, the world IMR declined from 126 in 1960 to 57 in 2001.^[48] However, IMR was, and remains, higher in LDCs. In 2001, the IMR for LDCs (91) was about 15 times as large as it was for MDCs (6). On average, for LDCs, the IMR is 17 times as higher than that of MDCs. Also, while both LDCs and MDCs made significant reductions in infant mortality rates, reductions among less developed countries are, on average, much less than those among the more developed countries.^[49]

A factor of about 67 separate countries with the highest and lowest reported infant mortality rates. The top and bottom five countries by this measure (taken from *The World Factbook's* 2012 estimates^[49]) are shown below.

Rank	Country	Infant mortality rate (deaths/1,000 live births)
1	 Afghanistan	121.63
2	 Niger	103.98
3	 Mali	103.08
4	 Sierra Leone	103.73
5	 Central African Republic	91.17
218	 Sweden	2.74
219	 Singapore	2.66
220	 Bermuda	2.47
221	 Japan	2.21
222	 Morocco	1.66

According to Gulbot, Gerland, Polekser and Saabneh "birth histories, however, are subject to a number of errors, including omission of deaths and age misreporting errors."^[50]

United States

Infant mortality rate in the US decreased by 2.3% to a historic low of 562 infant deaths per 100,000 live births in 2014.^[51]

Of the 27 most developed countries, the U.S. has the highest Infant Mortality Rate, despite spending much more on health care per capita^[52]^[53]^[54]^[55]^[56]^[57]^[58]^[59]^[60]^[61]^[62]^[63]^[64]^[65]^[66]^[67]^[68]^[69]^[70]^[71]^[72]^[73]^[74]^[75]^[76]^[77]^[78]^[79]^[80]^[81]^[82]^[83]^[84]^[85]^[86]^[87]^[88]^[89]^[90]^[91]^[92]^[93]^[94]^[95]^[96]^[97]^[98]^[99]^[100]^[101]^[102]^[103]^[104]^[105]^[106]^[107]^[108]^[109]^[110]^[111]^[112]^[113]^[114]^[115]^[116]^[117]^[118]^[119]^[120]^[121]^[122]^[123]^[124]^[125]^[126]^[127]^[128]^[129]^[130]^[131]^[132]^[133]^[134]^[135]^[136]^[137]^[138]^[139]^[140]^[141]^[142]^[143]^[144]^[145]^[146]^[147]^[148]^[149]^[150]^[151]^[152]^[153]^[154]^[155]^[156]^[157]^[158]^[159]^[160]^[161]^[162]^[163]^[164]^[165]^[166]^[167]^[168]^[169]^[170]^[171]^[172]^[173]^[174]^[175]^[176]^[177]^[178]^[179]^[180]^[181]^[182]^[183]^[184]^[185]^[186]^[187]^[188]^[189]^[190]^[191]^[192]^[193]^[194]^[195]^[196]^[197]^[198]^[199]^[200]^[201]^[202]^[203]^[204]^[205]^[206]^[207]^[208]^[209]^[210]^[211]^[212]^[213]^[214]^[215]^[216]^[217]^[218]^[219]^[220]^[221]^[222]^[223]^[224]^[225]^[226]^[227]^[228]^[229]^[230]^[231]^[232]^[233]^[234]^[235]^[236]^[237]^[238]^[239]^[240]^[241]^[242]^[243]^[244]^[245]^[246]^[247]^[248]^[249]^[250]^[251]^[252]^[253]^[254]^[255]^[256]^[257]^[258]^[259]^[260]^[261]^[262]^[263]^[264]^[265]^[266]^[267]^[268]^[269]^[270]^[271]^[272]^[273]^[274]^[275]^[276]^[277]^[278]^[279]^[280]^[281]^[282]^[283]^[284]^[285]^[286]^[287]^[288]^[289]^[290]^[291]^[292]^[293]^[294]^[295]^[296]^[297]^[298]^[299]^[300]^[301]^[302]^[303]^[304]^[305]^[306]^[307]^[308]^[309]^[310]^[311]^[312]^[313]^[314]^[315]^[316]^[317]^[318]^[319]^[320]^[321]^[322]^[323]^[324]^[325]^[326]^[327]^[328]^[329]^[330]^[331]^[332]^[333]^[334]^[335]^[336]^[337]^[338]^[339]^[340]^[341]^[342]^[343]^[344]^[345]^[346]^[347]^[348]^[349]^[350]^[351]^[352]^[353]^[354]^[355]^[356]^[357]^[358]^[359]^[360]^[361]^[362]^[363]^[364]^[365]^[366]^[367]^[368]^[369]^[370]^[371]^[372]^[373]^[374]^[375]^[376]^[377]^[378]^[379]^[380]^[381]^[382]^[383]^[384]^[385]^[386]^[387]^[388]^[389]^[390]^[391]^[392]^[393]^[394]^[395]^[396]^[397]^[398]^[399]^[400]^[401]^[402]^[403]^[404]^[405]^[406]^[407]^[408]^[409]^[410]^[411]^[412]^[413]^[414]^[415]^[416]^[417]^[418]^[419]^[420]^[421]^[422]^[423]^[424]^[425]^[426]^[427]^[428]^[429]^[430]^[431]^[432]^[433]^[434]^[435]^[436]^[437]^[438]^[439]^[440]^[441]^[442]^[443]^[444]^[445]^[446]^[447]^[448]^[449]^[450]^[451]^[452]^[453]^[454]^[455]^[456]^[457]^[458]^[459]^[460]^[461]^[462]^[463]^[464]^[465]^[466]^[467]^[468]^[469]^[470]^[471]^[472]^[473]^[474]^[475]^[476]^[477]^[478]^[479]^[480]^[481]^[482]^[483]^[484]^[485]^[486]^[487]^[488]^[489]^[490]^[491]^[492]^[493]^[494]^[495]^[496]^[497]^[498]^[499]^[500]^[501]^[502]^[503]^[504]^[505]^[506]^[507]^[508]^[509]^[510]^[511]^[512]^[513]^[514]^[515]^[516]^[517]^[518]^[519]^[520]^[521]^[522]^[523]^[524]^[525]^[526]^[527]^[528]^[529]^[530]^[531]^[532]^[533]^[534]^[535]^[536]^[537]^[538]^[539]^[540]^[541]^[542]^[543]^[544]^[545]^[546]^[547]^[548]^[549]^[550]^[551]^[552]^[553]^[554]^[555]^[556]^[557]^[558]^[559]^[560]^[561]^[562]^[563]^[564]^[565]^[566]^[567]^[568]^[569]^[570]^[571]^[572]^[573]^[574]^[575]^[576]^[577]^[578]^[579]^[580]^[581]^[582]^[583]^[584]^[585]^[586]^[587]^[588]^[589]^[590]^[591]^[592]^[593]^[594]^[595]^[596]^[597]^[598]^[599]^[600]^[601]^[602]^[603]^[604]^[605]^[606]^[607]^[608]^[609]^[610]^[611]^[612]^[613]^[614]^[615]^[616]^[617]^[618]^[619]^[620]^[621]^[622]^[623]^[624]^[625]^[626]^[627]^[628]^[629]^[630]^[631]^[632]^[633]^[634]^[635]^[636]^[637]^[638]^[639]^[640]^[641]^[642]^[643]^[644]^[645]^[646]^[647]^[648]^[649]^[650]^[651]^[652]^[653]^[654]^[655]^[656]^[657]^[658]^[659]^[660]^[661]^[662]^[663]^[664]^[665]^[666]^[667]^[668]^[669]^[670]^[671]^[672]^[673]^[674]^[675]^[676]^[677]^[678]^[679]^[680]^[681]^[682]^[683]^[684]^[685]^[686]^[687]^[688]^[689]^[690]^[691]^[692]^[693]^[694]^[695]^[696]^[697]^[698]^[699]^[700]^[701]^[702]^[703]^[704]^[705]^[706]^[707]^[708]^[709]^[710]^[711]^[712]^[713]^[714]^[715]^[716]^[717]^[718]^[719]^[720]^[721]^[722]^[723]^[724]^[725]^[726]^[727]^[728]^[729]^[730]^[731]^[732]^[733]^[734]^[735]^[736]^[737]^[738]^[739]^[740]^[741]^[742]^[743]^[744]^[745]^[746]^[747]^[748]^[749]^[750]^[751]^[752]^[753]^[754]^[755]^[756]^[757]^[758]^[759]^[760]^[761]^[762]^[763]^[764]^[765]^[766]^[767]^[768]^[769]^[770]^[771]^[772]^[773]^[774]^[775]^[776]^[777]^[778]^[779]^[780]^[781]^[782]^[783]^[784]^[785]^[786]^[787]^[788]^[789]^[790]^[791]^[792]^[793]^[794]^[795]^[796]^[797]^[798]^[799]^[800]^[801]^[802]^[803]^[804]^[805]^[806]^[807]^[808]^[809]^[810]^[811]^[812]^[813]^[814]^[815]^[816]^[817]^[818]^[819]^[820]^[821]^[822]^[823]^[824]^[825]^[826]^[827]^[828]^[829]^[830]^[831]^[832]^[833]^[834]^[835]^[836]^[837]^[838]^[839]^[840]^[841]^[842]^[843]^[844]^[845]^[846]^[847]^[848]^[849]^[850]^[851]^[852]^[853]^[854]^[855]^[856]^[857]^[858]^[859]^[860]^[861]^[862]^[863]^[864]^[865]^[866]^[867]^[868]^[869]^[870]^[871]^[872]^[873]^[874]^[875]^[876]^[877]^[878]^[879]^[880]^[881]^[882]^[883]^[884]^[885]^[886]^[887]^[888]^[889]^[890]^[891]^[892]^[893]^[894]^[895]^[896]^[897]^[898]^[899]^[900]^[901]^[902]^[903]^[904]^[905]^[906]^[907]^[908]^[909]^[910]^[911]^[912]^[913]^[914]^[915]^[916]^[917]^[918]^[919]^[920]^[921]^[922]^[923]^[924]^[925]^[926]^[927]^[928]^[929]^[930]^[931]^[932]^[933]^[934]^[935]^[936]^[937]^[938]^[939]^[940]^[941]^[942]^[943]^[944]^[945]^[946]^[947]^[948]^[949]^[950]^[951]^[952]^[953]^[954]^[955]^[956]^[957]^[958]^[959]^[960]^[961]^[962]^[963]^[964]^[965]^[966]^[967]^[968]^[969]^[970]^[971]^[972]^[973]^[974]^[975]^[976]^[977]^[978]^[979]^[980]^[981]^[982]^[983]^[984]^[985]^[986]^[987]^[988]^[989]^[990]^[991]^[992]^[993]^[994]^[995]^[996]^[997]^[998]^[999]^[1000]^[1001]^[1002]^[1003]^[1004]^[1005]^[1006]^[1007]^[1008]^[1009]^[1010]^[1011]^[1012]^[1013]^[1014]^[1015]^[1016]^[1017]^[1018]^[1019]^[1020]^[1021]^[1022]^[1023]^[1024]^[1025]^[1026]^[1027]^[1028]^[1029]^[1030]^[1031]^[1032]^[1033]^[1034]^[1035]^[1036]^[1037]^[1038]^[1039]^[1040]^[1041]^[1042]^[1043]^[1044]^[1045]^[1046]^[1047]^[1048]^[1049]^[1050]^[1051]^[1052]^[1053]^[1054]^[1055]^[1056]^[1057]^[1058]^[1059]^[1060]^[1061]^[1062]^[1063]^[1064]^[1065]^[1066]^[1067]^[1068]^[1069]^[1070]^[1071]^[1072]^[1073]^[1074]^[1075]^[1076]^[1077]^[1078]^[1079]^[1080]^[1081]^[1082]^[1083]^[1084]^[1085]^[1086]^[1087]^[1088]^[1089]^[1090]^[1091]^[1092]^[1093]^[1094]^[1095]^[1096]^[1097]^[1098]^[1099]^[1100]^[1101]^[1102]^[1103]^[1104]^[1105]^[1106]^[1107]^[1108]^[1109]^[1110]^[1111]^[1112]^[1113]^[1114]^[1115]^[1116]^[1117]^[1118]^[1119]^[1120]^[1121]^[1122]^[1123]^[1124]^[1125]^[1126]^[1127]^[1128]^[1129]^[1130]^[1131]^[1132]^[1133]^[1134]^[1135]^[1136]^[1137]^[1138]^[1139]^[1140]^[1141]^[1142]^[1143]^[1144]^[1145]^[1146]^[1147]^[1148]^[1149]^[1150]^[1151]^[1152]^[1153]^[1154]^[1155]^[1156]^[1157]^[1158]^[1159]^[1160]^[1161]^[1162]^[1163]^[1164]^[1165]^[1166]^[1167]^[1168]^[1169]^[1170]^[1171]^[1172]^[1173]^[1174]^[1175]^[1176]^[1177]^[1178]^[1179]^[1180]^[1181]^[1182]^[1183]^[1184]^[1185]^[1186]^[1187]^[1188]^[1189]^[1190]^[1191]^[1192]^[1193]^[1194]^[1195]^[1196]^[1197]^[1198]^[1199]^[1200]^[1201]^[1202]^[1203]^[1204]^[1205]^[1206]^[1207]^[1208]^[1209]^[1210]^[1211]^[1212]^[1213]^[1214]^[1215]^[1216]^[1217]^[1218]^[1219]^[1220]^[1221]^[1222]^[1223]^[1224]^[1225]^[1226]^[1227]^[1228]^[1229]^[1230]^[1231]^[1232]^[1233]^[1234]^[1235]^[1236]^[1237]^[1238]^[1239]^[1240]^[1241]^[1242]^[1243]^[1244]^[1245]^[1246]^[1247]^[1248]^[1249]^[1250]^[1251]^[1252]^[1253]^[1254]^[1255]^[1256]^[1257]^[1258]^[1259]^[1260]^[1261]^[1262]^[1263]^[1264]^[1265]^[1266]^[1267]^[1268]^[1269]^[1270]^[1271]^[1272]^[1273]^[1274]^[1275]^[1276]^[1277]^[1278]^[1279]^[1280]^[1281]^[1282]^[1283]^[1284]^[1285]^[1286]^[1287]^[1288]^[1289]^[1290]^[1291]^[1292]^[1293]^[1294]^[1295]^[1296]^[1297]^[1298]^[1299]^[1300]^[1301]^[1302]^[1303]^[1304]^[1305]^[1306]^[1307]^[1308]^[1309]^[1310]^[1311]^[1312]^[1313]^[1314]^[1315]^[1316]^[1317]^[1318]^[1319]^[1320]^[1321]^[1322]^[1323]^[1324]^[1325]^[1326]^[1327]^[1328]^{[1}

Wikipedia on infant mortality

W Infant mortality - Wikipedia x +

← → ↻ https://en.wikipedia.org/wiki/Infant_mortality

Epidemiology [edit]

See also: *List of countries by infant mortality rate*

For the world, and for both less developed countries (LDCs) and more developed countries (MDCs), IMR declined. However, IMR was, and remains, higher in LDCs. In 2001, the IMR for LDCs (91) was about 10 times as large as that for MDCs. However, on average, much less than those among the more developed countries. ^[*clarification needed*]

A factor of about 67 separates the countries with the highest and lowest reported infant mortality rates. The top and

Rank	Country	Infant mortality rate (deaths/1,000 live births)
1	Afghanistan	121.63
2	Niger	109.98
3	Mali	109.08
4	Somalia	103.72
5	Central African Republic	97.17
218	Sweden	2.74
219	Singapore	2.65
220	Bermuda	2.47
221	Japan	2.21
222	Monaco	1.80

According to Guillot, Gerland, Pelletier and Saabneh "birth histories, however, are subject to a number of error

Fetching a table from Wikipedia

```
library(rvest)

# 1) fetch and parse the website
page <- read_html("https://en.wikipedia.org/wiki/Infant_mortality")
# 2) extract the html node containing the table
table <- html_node(page,
                    xpath = "//*[@id='mw-content-text']/div/table[2]")
# 3) extract the table as a data frame
mrates <- html_table(table)
```

Inspecting the HTML source

- Convenient with modern browsers: Use the developer tools (right-click *Inspect*).
- Look at the HTML source to grasp the structure.
- Find out how to navigate the site.
- Find the element(s) you want to extract.
- Get the Xpath expression or CSS selector to extract elements.

Infant mortality rates from Wikipedia

```
<table class="wikitable" style="text-align:left">
  <tbody>
    <tr>
      <th>Rank</th>
      <th>Country</th>
      <th>Infant mortality rate <br> (deaths/1,000 live births)</th>
    </tr>
    <tr>
      <td>1</td>
      <td style="text-align:left;"><a href="/wiki/Afghanistan" title="Afghanistan">Afghanistan</a></td>
      <td>121.63</td>
    </tr>
    <tr>
      <td>2</td>
      <td style="text-align:left;"><a href="/wiki/Niger" title="Niger">Niger</a></td>
      <td>109.98</td>
    </tr>
    <tr>
      <td>3</td>
      <td style="text-align:left;"><a href="/wiki/Mali" title="Mali">Mali</a></td>
      <td>109.08</td>
    </tr>
    <tr>
      <td>4</td>
      <td style="text-align:left;"><a href="/wiki/Somalia" title="Somalia">Somalia</a></td>
      <td>103.72</td>
    </tr>
    ...
  </tbody>
</table>
```


CSS selectors and XPath expressions

```
# fetch and parse the website
page <- read_html("https://en.wikipedia.org/wiki/Infant_mortality")
# list the table nodes
html_nodes(page, "table")
# using xpath expressions or css selectors is equivalent
table <- html_node(page,
                    xpath = "//*[@id='mw-content-text']/div/table[2]")
table <- html_node(page,
                    css = "#mw-content-text > div > table:nth-child(121)")
```

Examples

The general structure

- There is no universal recipe. But most programs follow a certain structure.
 1. Open a website mimicking a browser and navigate it (optional).
 2. Get the page source HTML and feed it to a parser.
 3. Extract the elements you need.
 4. Filter and arrange them as needed and save them.
 5. Repeat 1.-4. as needed.

Navigating to another page

```
# Open infant mortality page
session <- html_session("https://en.wikipedia.org/wiki/Infant_mortality")
# Goto page on Somalia
session <- follow_link(session, "Somalia")
# Read the source
page <- read_html(session)
# Extract html
table <- html_node(page,
                    xpath = "//*[@id='mw-content-text']/div/table[4]")
regions <- html_table(table)
```

Filtering links

```
# read wiki page
page <- read_html("https://en.wikipedia.org/wiki/Infant_mortality")
# get the links
wikilinks <- html_attr(html_nodes(page, "a"), "href")
# use regex to filter internal links:
#   select only articles, no files or category pages,
#   matching with mortality or somalia
links <- grep("^(?!.*:)(/wiki/. *Mortality)|(/wiki/. *Somalia)", wikilinks,
              ignore.case = TRUE, value = TRUE, perl = TRUE)
links <- unique(links)
# go to first selected article page and process it
session <- jump_to(session, links[1])
page <- read_html(session)
html_nodes(page, "title")
```

A more elaborate example

- Phillippine Statistics Authority *Good Governance Index*.
- Available at <http://nap.psa.gov.ph/ggi/default.asp>.
- Available at <https://web.archive.org/web/20190915135458/http://nap.psa.gov.ph/ggi/default.asp>.
- Get all GGI data tables for all municipalities.
- Save them in a local data file for further analysis.
- How would you go about this?

If simple navigation fails

- Some web pages cannot be navigated easily with simpler requests.
- Often due to hidden Javascript or other server-side processing.
- In this case, resort to Selenium (`library(rselenium)` in R).
- Under the hood, Selenium relies on a complete browser running in a container.
- Slower and comes with substantial overhead costs.
- Only use when absolutely necessary.

Another example

- WHO venomous snakes distribution and species risk categories
- Database form link:
<https://apps.who.int/bloodproducts/snakeantivenoms/database/SearchFrm.aspx>
- Collect snake data for all countries.
- Getting dropdown options and initial form submission straightforward.
- So is table extraction.
- Navigating further links is tricky.

General remarks

- Start simple and expand your program incrementally.
- Keep it simple. Do not overengineer the problem. Do not repeat yourself.
- Limit the number of iterations for test runs. Use print statements to inspect objects.
- Write tests to verify things work as intended.
- If your program requires complex monitoring/validation procedures or threading for performance, use Python.

Final remarks

- Sometimes small programs can go a long way.
- Do not lose sight of your ultimate goal. Time is valuable.
- Do not engage in perfectionism, focus on GTD.
- Identify everyday tasks that you can optimize.
- It might even be fun.

Assignment

Assignment, part I

Choose *one* of the following.

Open Data Tanzania

<https://tanzania.opendataforafrica.org/>

- ➡ For each region in Tanzania, get the basic facts (top of the page: capital, population, area, ...) and mortality statistics (table under link 'Deaths').

WHO Venomous Snakes Database

<http://apps.who.int/bloodproducts/snakeantivenoms/database/SearchFrm.aspx>

- ➡ For every country, collect *all* species of venomous snakes.

Swiss Tax Calculator

<https://swisstaxcalculator.estv.admin.ch/#/taxburden/income-wealth-tax>

- ➡ Collect municipal income tax burden statistics for the following parameters: All payers, atheist, geographical comparison (all municipalities), all years, income 50'000 CHF.

Assignment, part I

- Submission deadline is November 6, 2020.
- Submit code only, no data.
- Comment your code or submit a short description alongside.
- A proof-of-concept restricted to the first couple of regions/... is fine.
- Accounts for 20% of the final grade.

Next lecture: Text as data.