

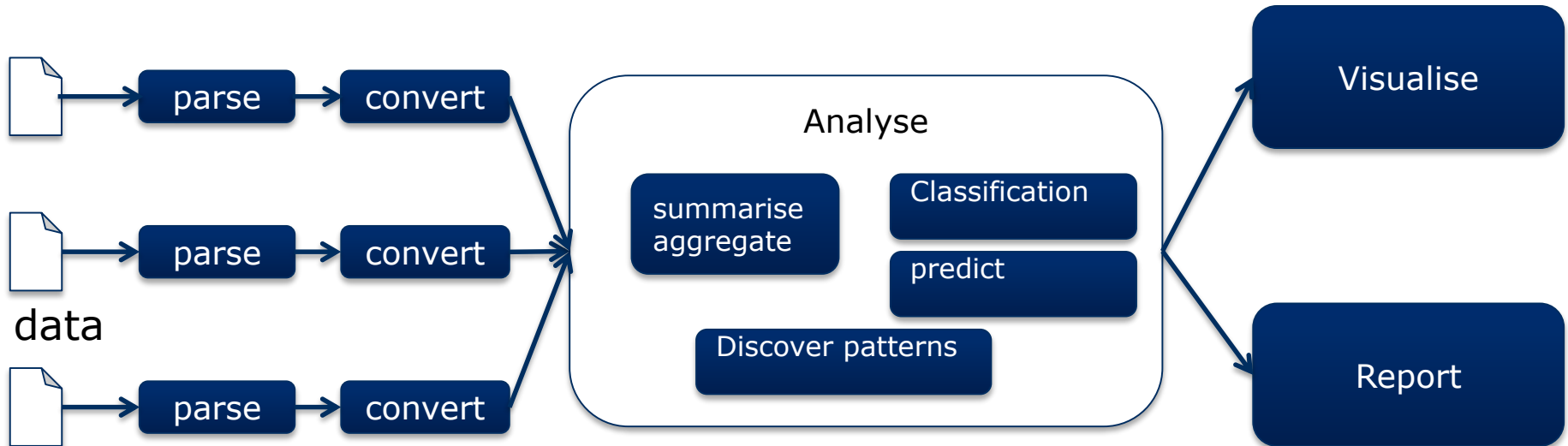
What can you do with the data

- Data processing
Provide/filter data
- Visualisations
charts, graphs, maps, ...



From data to “meaning”

- Parsing, converting, analysing, representing



Parsing File Formats

Tabular, Tree, Graph, Text
 How to parse the data?

Lorem ipsum dolor si Integer aliquam sem vitae ipsum vehicula e fringilla

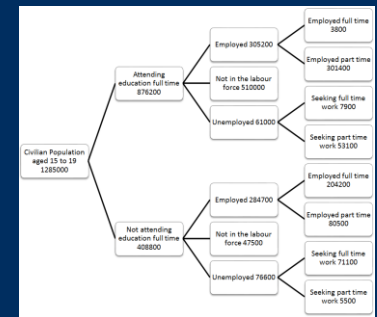
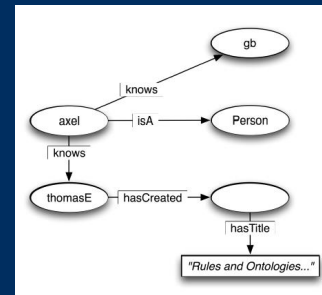
Duocet et nisi lorem, sed rhoncus odio. Pellentesque ut nulla, conmodo id accumsan ac, volutpat quis ligula. Nulla libero felis, venenatis id varius in, vehicula eu lectus. Suspendisse porttitor sed in massa. Luctus viverra interdum eros hendrerit, porttitor volutpat quis.

Mauris mollis ac consequat vulputate. Duocet dignissim tempus suscipit. Sed tempus malesuada mollis. Etiam ullamcorper, orn et vitae blandit malesuada, habitus orn consequat dui, id adipiscing magna quam eu felis. Nunc at amet turpis nisi. Cras nulla turpis, imperdiet non hac habitant etiam, ullamcorper nec ligula. Ut lacinia, risus at amet sodales cursus,

ne sit amet turpis nisi. Cras nulla turpis, imperdiet non hendrerit vitae, ullamcorper nec ligula. Ut lacinia, risus at amet sodales cursus, ugnem felis gravida nulla, ullamcorper dignissim turpis lacin sed temp. Duocet nisi nisi, fringilla eget aliquet sollicitudin, suscipit eu nulla. Suspendisse vitae risus lacinia, eget ornato. Lacinia tristique eget aliquet sollicitudin.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum id odio lorem, in bibendum orn. Integer nunc nunc tristique aliquet. Suspendisse eget magna vitae.

	A	B	C
1	roomcode	category_en	capacity
2	LC.0.000	Event space	1500
3	TC.0.10	Event space	650
4	LC.0.110	Event space	400
5	SC.2.733	Event space	200
6	LC.0.132	Event space	200
7	TC.0.02	Auditorium	180
8	TC.0.01	Auditorium	180
9	TC.0.03	Auditorium	180
10	TC.0.04	Auditorium	180
11	TC.1.02	Auditorium	120
12	TC.1.01	Auditorium	120
13			



Different Formats

A N W E S E N D E .

VORSITZENDE: Bürgermeister Mag. Siegfried NAGL
 Gemeinderätin Gerda GESEK

Weiters 48 Mitglieder des Gemeinderates, und zwar:

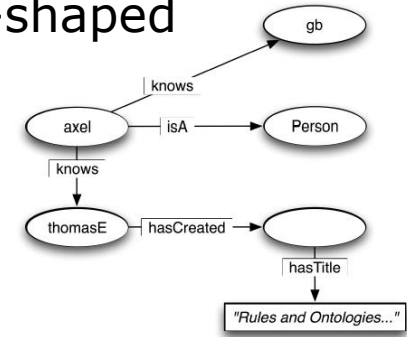
BERGMANN Ingeborg
 BRAUNERSREUTHER Christine
 DREISIEBNER Karl
 EBER Manfred
 FABISCH Andreas Mag.
 FRÖLICH Klaus Mag.

plain text

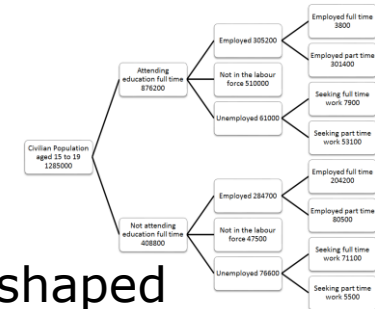
tabular

	A	B	C
1	roomcode	category_en	capacity
2	LC.0.000	Event space	1500
3	TC.0.10	Event space	650
4	LC.0.110	Event space	400
5	SC.2.733	Event space	200
6	LC.0.132	Event space	200
7	TC.0.02	Auditorium	180
8	TC.0.01	Auditorium	180
9	TC.0.03	Auditorium	180
10	TC.0.04	Auditorium	180
11	TC.1.02	Auditorium	120
12	TC.1.01	Auditorium	120
13			

graph-shaped



tree-shaped



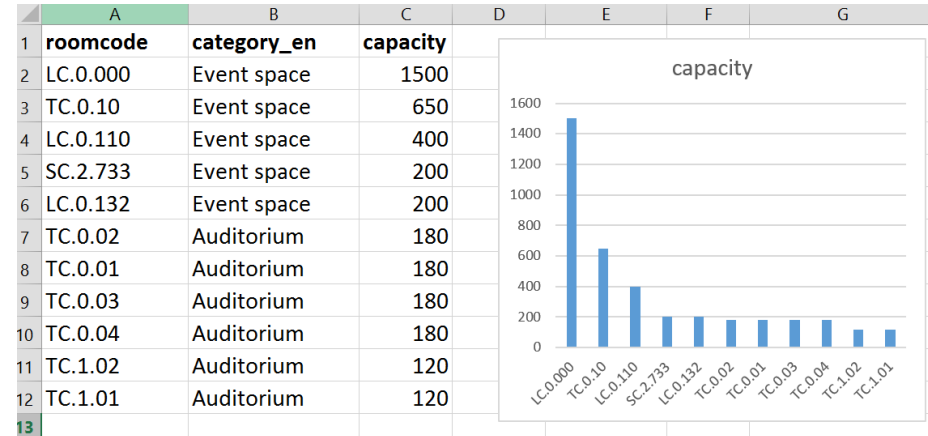
■ CSV

- Comma-separated values
- Plain text
- Variations on the format

```
course_id,semester,name
4000,15S,Marketing
4001,15S,Marketing
4002,15S,"Personal, Führung, Organisation"
4004,15S,Grundlagen der Volkswirtschaftslehre
4006,15S,Mathematik
4007,15S,Statistik
```

■ Excel

- .xls, .xlsx, .xlsm, ... (binary)
- Widely used but proprietary
- Built-in visualisation



- Java: [Commons CSV](#)

```
File csvData = new File("/path/to/csv");
CSVParser parser = CSVParser.parse(csvData, CSVFormat.RFC4180);
for (CSVRecord csvRecord : parser) {
    ...
}
```

Common CSV

- Python:

- **CSV:** standard library
- [messytables](#): delimiter detection, datatype detection, ...

- JavaScript:

- [Papa Parse](#): JSON conversion, delimiter detection

```
var results = Papa.parse(csvString);
console.log(results.meta.delimiter);
```

Papa Parse

- [CSVKit](#):

- Command line tool, Python lib
- grep, clean, processing

Tree-based: Hierarchies

■ XML

- Markup language
- Tag based (not predefined like HTML)
- Human readable

```
<Gemeindedaten>
  <Gemeinde>
    <bgmname>Mag. Siegfried NAGL</bgmname>
    <email>stadtverwaltung@graz.at</email>
    <gemfax>+43 (316) 872-2369</gemfax>
    <gemname>Graz</gemname>
    <gemtel>+43 (316) 872-0</gemtel>
    <ort>Graz</ort>
    <plz>8010</plz>
    <strasse>Hauptplatz 1</strasse>
    <vbgmname>Mag.a Dr.in Martina SCHRÖCK</vbgmname>
    <webadr>http://www.graz.at</webadr>
  </Gemeinde>
</Gemeindedaten>
```

■ JSON

- **JavaScript Object Notation**
- Identical to JavaScript objects
- Name/value pairs
- Easy-to-use alternative to XML

```
{
  id: "UNIVERSITAETOGD.762955",
  geometry: {
    type: "Point",
    coordinates: [
      16.408860446502068,
      48.21384341856702
    ]
  },
  properties: {
    NAME: "Wirtschaftsuniversität Wien",
    BEZEICHNUNG: "LC - Hauptgebäude"
  }
}
```

- Java:
 - DOM parser
 - SAX parser: event based, no file size limits
- Python:
 - ElementTree: simple-to-use

```
import xml.etree.ElementTree as ET
tree = ET.parse('myfile.xml')
root = tree.getroot()

for child in root:
    print child.tag, child.attrib
```

ElementTree

- Command line:
 - xmllint: validation (against a DTD)

```
user@users-desktop:~$ xmllint --valid --noout gem.xml
gem.xml:1: validity error : Validation failed: no DTD found !
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><Gemeindedaten><Gemeinde>
                                     ^
user@users-desktop:~$
```

xmllint

- Keyhole Markup Language
- XML-based schema for geographic data
- Default XML parser
- Java:
 - [JAK](#)
- Python:
 - [pyKML](#): validation, KML construction

```
final Kml kml = Kml.unmarshal(new File("..."));  
final Placemark placemark = (Placemark) kml.getFeature();  
Point point = (Point) placemark.getGeometry();  
List<Coordinate> coordinates = point.getCoordinates();  
for (Coordinate coordinate : coordinates) {  
    System.out.println(coordinate.getLatitude());  
    System.out.println(coordinate.getLongitude());  
    System.out.println(coordinate.getAltitude());  
}
```

JAK

■ Java:

- org.json: simple-to-use
- [Gson](https://gson.org): serialize java classes, support for java generics

```
String str = "{ \"name\": \"Alice\", \"age\": 20 }";
JSONObject obj = new JSONObject(str);
String n = obj.getString("name");
int a = obj.getInt("age");
System.out.println(n + " " + a); // prints "Alice 20" org.json
```

■ Python:

- [json](https://python.org): standard library

```
import json
with open("../") as json_file:
    # json.load for file object arguments
    json_data = json.load(json_file)

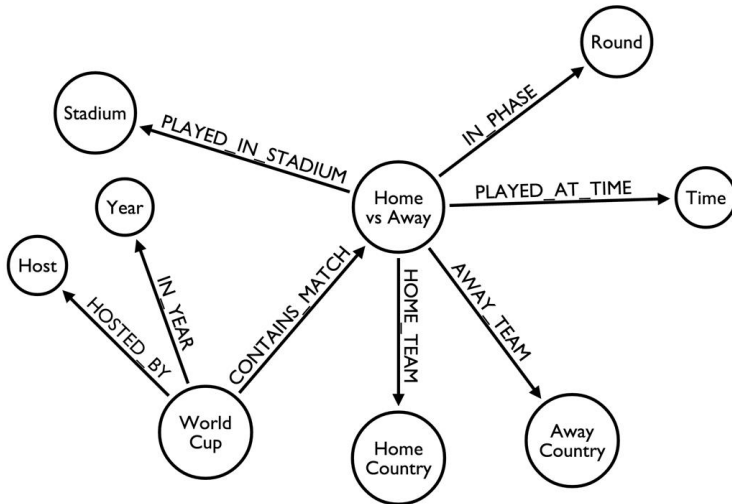
# json.loads for string arguments
json_data = json.loads("{ \"name\": \"Alice\", \"age\": 20 }")
print json_data["name"] # prints Alice Python
```

■ Command line:

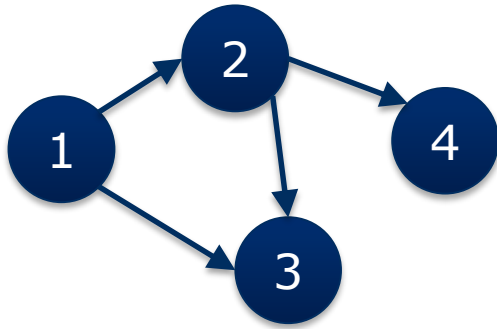
- [jq](https://stedolan.github.io/jq/)

```
user@users-desktop:~$ curl http://data.wu.ac.at/api/rest/dataset/all_course_events_2015s |
| jq '.title'
"All Course-Events during SS15 at WU Vienna"
user@users-desktop:~$ jq
```

Graph-shaped data formats



Graph representation



- Edge-list

1 2
1 3
2 3
2 4

- Adjacency-list

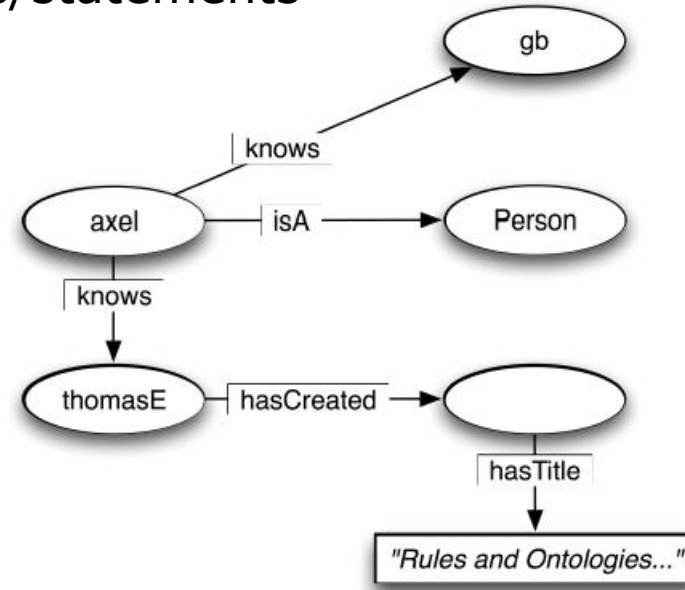
1: 2,3
2: 3,4

- Matrix

	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	0
4	0	1	0	0

- RDF (**R**esource **D**escription **F**ramework)
 - Describing resources per triples/statements
 - **S**ubject **P**redicate **O**bject

axel isA Person .
axel hasName "Axel Polleres".
axel knows gb .
axel knows thomas.
thomas hasCreated an Article
titled "Rules and Ontologies ...".



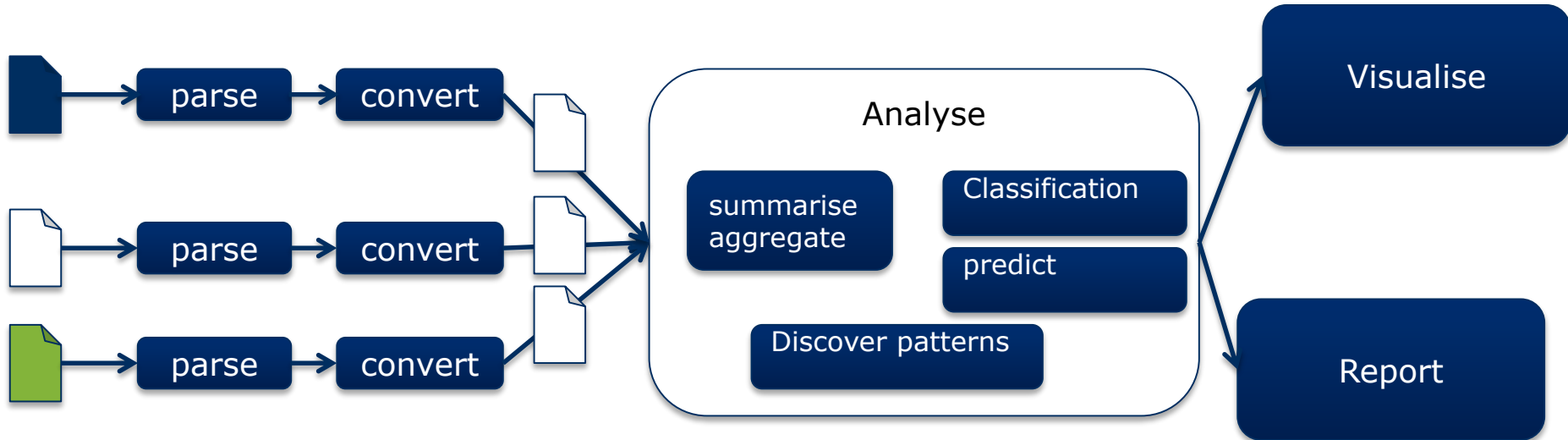
- Different syntaxes for RDF:
 - **RDF/XML**
 - Turtle
 - RDFa
 - JSON-LD
- Accessing RDF:
 - Query language: SPARQL
 - Parser

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/" >
  <foaf:Person rdf:ID="me">
    <foaf:name>Axel Polleres</foaf:name>
    <foaf:title>Dr</foaf:title>
    <foaf:firstName>Axel</foaf:firstName>
    <foaf:lastName>Polleres</foaf:lastName>

    <foaf:knows>
      <foaf:Person>
        <foaf:name>Thomas Eiter</foaf:name>
      </foaf:Person>
    </foaf:knows>
  </foaf:Person>
</rdf:RDF>
```

RDF/XML

Converting



Popular Functionality

JSON Validator	JSON Viewer	JSON Editor
XML to JSON	Encryption-Decryption	EXCEL TO HTML
	Javascript Validator	CSS Be

Web Viewer

JSON	XML	MXML
JavaScript	RSS	JAVA
	SQL	Online Editor

New Functionality

BASE64 To IMAGE CONVERTER	Send Snap Message	RGB TO HEX CONVERTER
HEX TO RGB CONVERTER	HEX TO CMYK CONVERTER	ONLINE XML EDITOR

CSV TO JSON

Save & Share

CSV Input

```
1 phone,os,size_inch,size_cm,ppi
2 Nokia Lumiya 1020,windows,4.5,11.0,332
3 Nokia Lumiya 520,windows,4.0,10.0,233
4 Nokia Lumiya 620,windows,3.8,9.7,247
5 Nokia Lumiya 720,windows,4.3,10.9,217
6 Nokia Lumiya 900,windows,4.3,11.0,217
```

EDITOR | CHECKER

Load Url

Browse

csv To json

Download

Result : CSV TO JSON

```
1 [
2 {
3   "phone": "Nokia Lumiya 1020",
4   "os": "windows",
5   "size_inch": "4.5",
6   "size_cm": "11.0",
7   "ppi": "332"
8 },
9 {
10  "phone": "Nokia Lumiya 520",
11  "os": "windows",
12  "size_inch": "4.0",
13  "size_cm": "10.0",
14  "ppi": "233"
15 },
16 {
17  "phone": "Nokia Lumiya 620",
18  "os": "windows",
19  "size_inch": "3.8",
20  "size_cm": "9.7",
21  "ppi": "247"
22 },
23 {
24  "phone": "Nokia Lumiya 720",
25  "os": "windows",
26  "size_inch": "4.3",
27  "size_cm": "10.9",
28  "ppi": "217"
29 },
30 {
31  "phone": "Nokia Lumiya 900",
32  "os": "windows",
```


Data processing: Some Buzzwords

- Validation
- Converting
- Sorting
- Summarization
- Aggregation
- Analysis
- Reporting
- Classification
- ...

E.g.: When to go to Mensa?

- Locate the data:

The screenshot shows a web interface for managing data. At the top right, there are two buttons: 'Manage' (with a wrench icon) and 'Download' (with a download icon). The main heading is 'All course-events SS15'. Below this, a URL is provided: <http://data.wu.ac.at/dataset/5649053b-44c2-4c5e-a24e-805c3b4d3be7/resource/a5b021e6-d16e-4d37-81df-ecc3b>. A descriptive text reads: 'A list of all Courses and their events at the Vienna University of Economics and Business in SS15'. Below the text, there are navigation options: 'Grid', 'Graph', and 'Map'. To the right, it says '15513 records' and a pagination control showing '0' of '100' records. A search bar is also present with the text 'Search data ...'. The main content is a table with 8 rows and 8 columns.

_id	course_id	semester	name	roomcode	building...	start	end
1	4000	2015-02-17T...	Marketing	TC.0.10 ...	TC	2015-03-1...	2015-03-1...
2	4000	2015-02-17T...	Marketing	TC.0.10 ...	TC	2015-03-1...	2015-03-1...
3	4000	2015-02-17T...	Marketing	TC.0.10 ...	TC	2015-03-2...	2015-03-2...
4	4000	2015-02-17T...	Marketing	TC.0.10 ...	TC	2015-04-1...	2015-04-1...
5	4001	2015-02-17T...	Marketing	TC.0.03 ...	TC	2015-05-1...	2015-05-1...
6	4001	2015-02-17T...	Marketing	TC.0.03 ...	TC	2015-05-2...	2015-05-2...
7	4001	2015-02-17T...	Marketing	TC.0.03 ...	TC	2015-06-0...	2015-06-0...
8	4001	2015-02-17T...	Marketing	TC.0.10 ...	TC	2015-06-1...	2015-06-1...

E.g.: When to go to Mensa?

- Process and filter:

```
import csv
from dateutil.parser import parse
from collections import Counter
import datetime

f = open('allcoursesandevents15s.csv', 'r')
csvf = csv.reader(f)

ends = []
for row in csvf:
    ends.append(row[6])
ends = ends[1:]
dates = [parse(e) for e in ends]

counts = Counter(dates)
now = datetime.datetime.now() + datetime.timedelta(days=5)
next_week = now + datetime.timedelta(days=20)

week = []
for date in counts:
    if now < date < next_week:
        if 11 <= date.hour <= 15:
            week.append((date, counts[date]))
sorted_week = sorted(week)

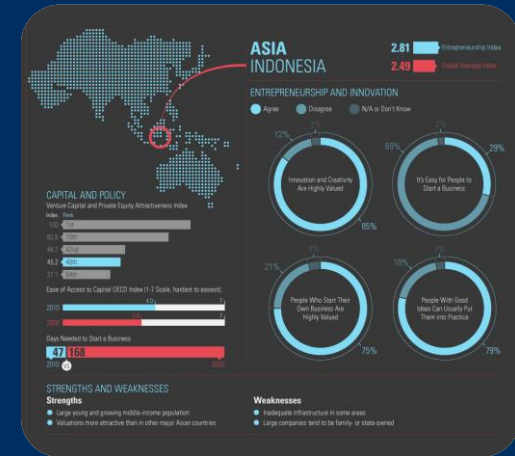
outfile = open('this_week_course_end.csv', 'w')
for date, count in sorted_week:
    outfile.write(date.strftime("%d.%m. %H:%M") + ',' + str(count) + '\n')
```


Visualisation

tell a story with your data

appealing presentation

pictures tell more than thousand words



Types of visualisation

- Charts
- Graphs
- Maps
- Infographics
- Interactive/complex Visualisations

Google Charts API

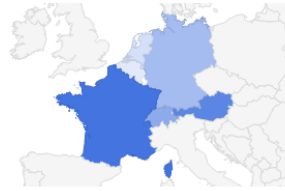
Hello, Charts!

- Quickstart
- Load the Libraries
- Prepare the Data
- Customize the Chart
- Draw the Chart

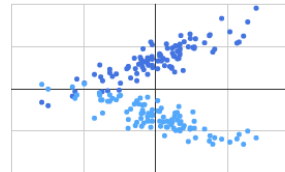
Chart Types

- [Chart Gallery](#)
- Annotation Charts
- Area Charts
- Bar Charts
- Bubble Charts
- Calendar Charts
- Candlestick Charts
- Column Charts
- Combo Charts
- Diff Charts
- Donut Charts
- Gauge Charts
- Geo Charts
- Histograms
- Intervals
- Line Charts
- Maps
- Org Charts
- Pie Charts
- Sankey Diagrams
- Scatter Charts
- Stepped Area Charts
- Table Charts
- Timelines
- Tree Map Charts

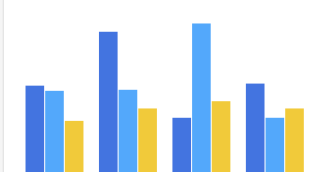
Geo Chart



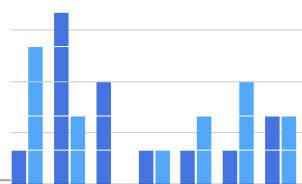
Scatter Chart



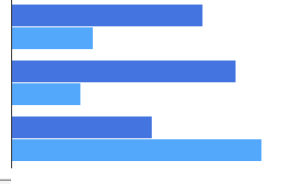
Column Chart



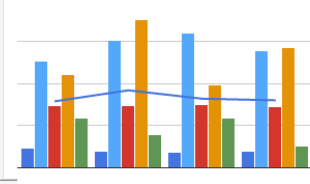
Histogram



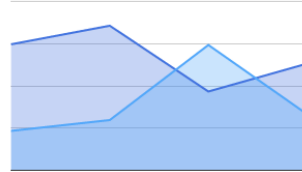
Bar Chart



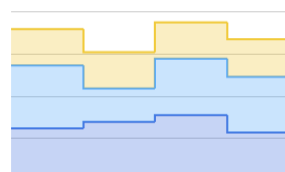
Combo Chart



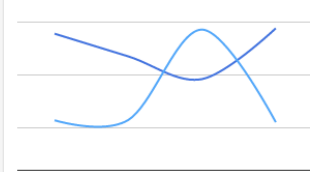
Area Chart



Stepped Area Chart



Line Chart



- [mapsdata](#)
 - Simple online tool
- JavaScript:
 - [OpenLayers](#)
 - [Leaflet](#)
- Python:
 - [basemap](#): matplotlib plugin
- [Google Maps API](#)
 - API Clients for JS, Java, Python, .NET, ...



Google Maps API

Lecture rooms on WU campus

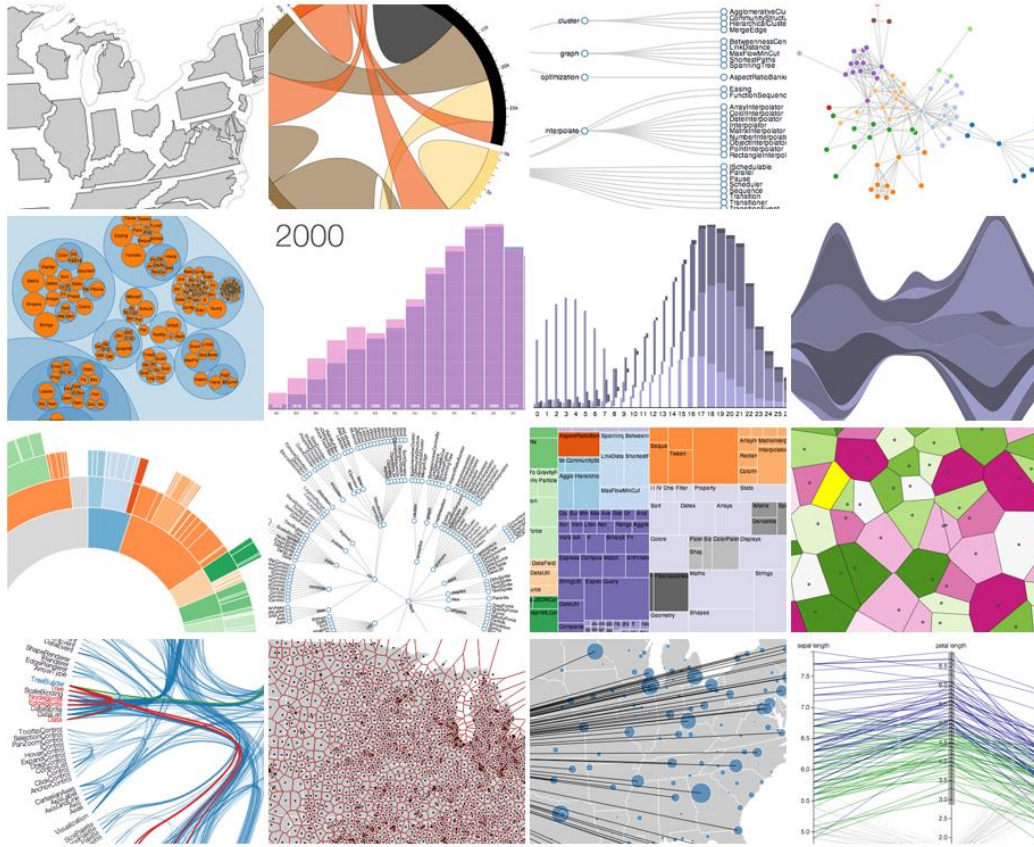
mapsdata

How to use Mapsdata

The screenshot displays the Mapsdata application interface. On the left is a sidebar with icons for Import, My Data, View, Search, Maps, Export, and Tweets. The main area shows a map of the WU campus with several buildings and streets. Red circles with numbers (1-17) are placed on the map to indicate lecture rooms. A control panel on the left side of the map includes:

- Import: Bubble (selected) and Cluster
- My Data: Heatmap and Markers
- View: A color palette with red, orange, green, purple, and blue segments.
- Search: A color palette with black, grey, green, brown, and blue segments.
- Opacity: A slider set to 60%.
- Distance: A slider set to 25px.
- Clear Map: A button to clear the map.

Graphs

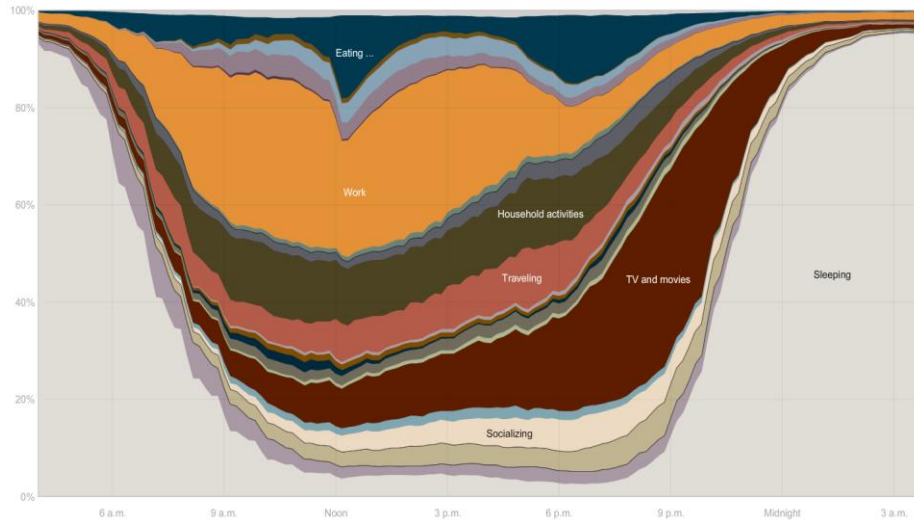


Visualising people's behaviour

Everyone

Sleeping, eating, working and watching television take up about two-thirds of the average day.

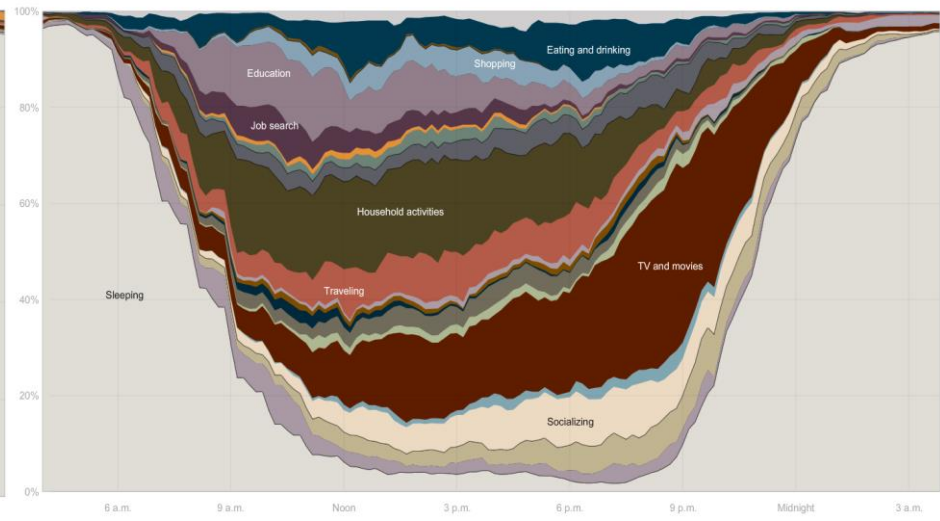
Everyone	Employed	White	Age 15-24	H.S. grads	No children
Men	Unemployed	Black	Age 25-64	Bachelor's	One child
Women	Not in lab...	Hispanic	Age 65+	Advanced	Two+ children



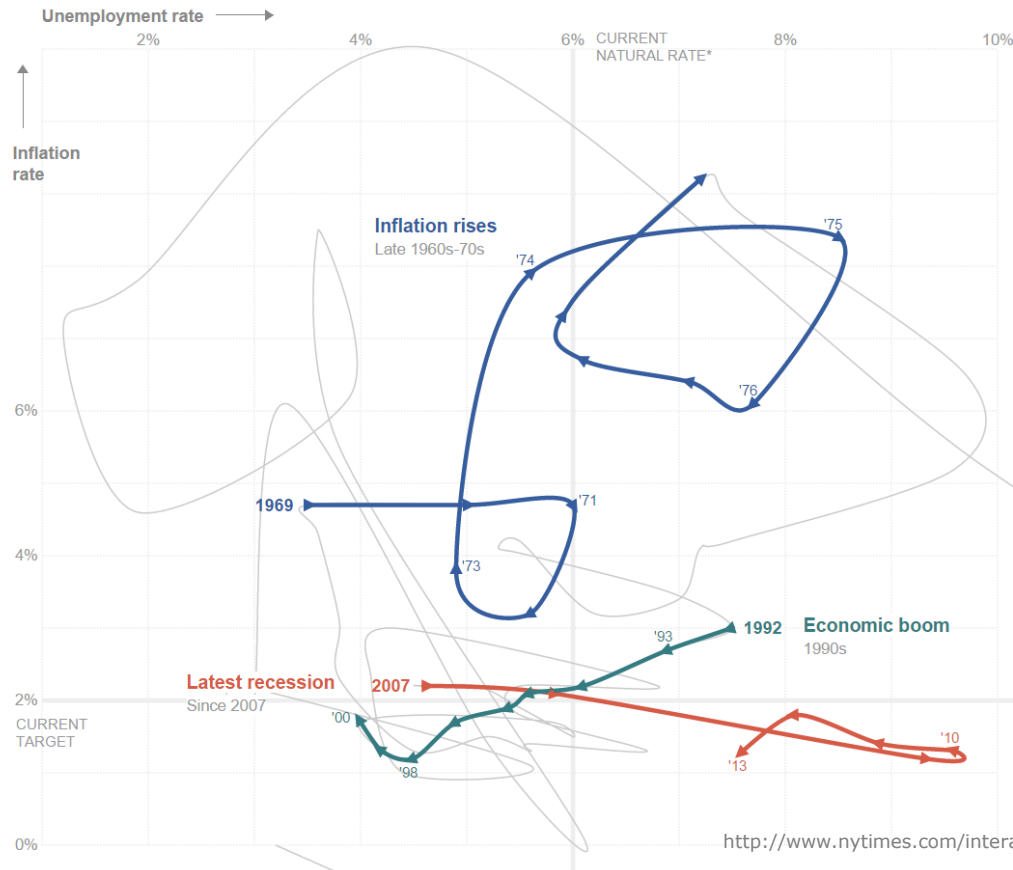
The unemployed

On average, the unemployed spend about a half-hour looking for work. They tidy the house, do laundry and yard work for more than two hours, about an hour more than the employed.

Everyone	Employed	White	Age 15-24	H.S. grads	No children
Men	Unemployed	Black	Age 25-64	Bachelor's	One child
Women	Not in lab...	Hispanic	Age 65+	Advanced	Two+ children



D3 (Data-Driven Documents)



<http://www.nytimes.com/interactive/2013/10/09/us/yellen-fed-chart.html>

- JavaScript:
 - [Flot](#): library for jQuery
 - [Raphaël](#)
 - [D3](#) (complex visualisations)
- Python:
 - [matplotlib](#)
 - [Seaborn](#)
- [Processing](#): JS and Python bindings
- [R](#): statistical package

Further Links

- <http://selection.datavisualization.ch/>
- http://www.dmoz.org/Science/Math/Combinatorics/Software/Graph_Drawing/
- Java Open Source Chart libs:
 - <http://www.fromdev.com/2012/09/Free-Open-Source-Java-Charting-Library.html>
- <http://socialcompare.com/en/comparison/javascript-graphs-and-charts-libraries>