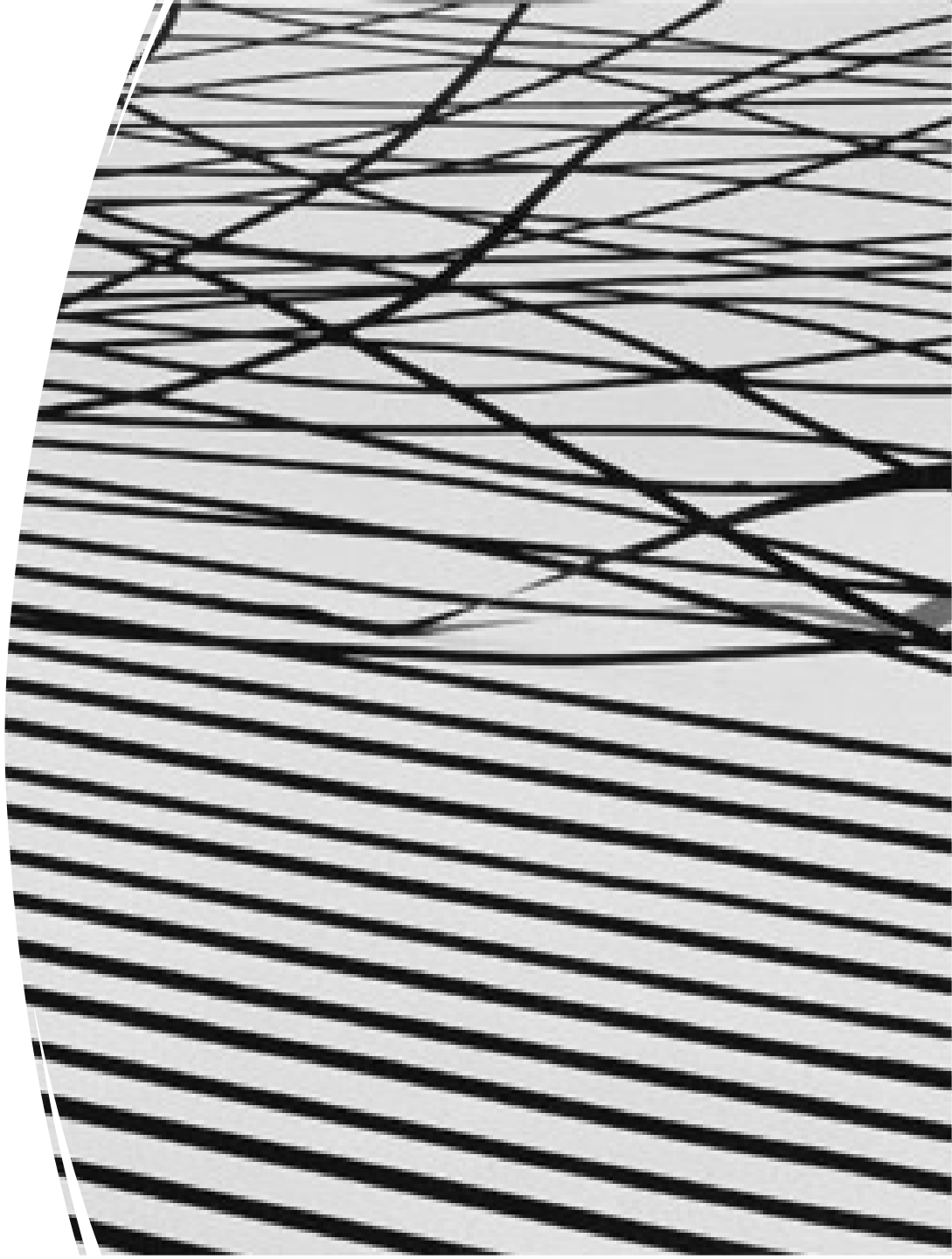
Structured Data

Databases in a broad sense



Agenda

- CSV/Tabular Data/Excel
- JSON
- JQ
- Discussion in Groups / Exercise

Tabular data (CSV/Excel)

- Often used for structuring research data
- We all have Excel or similar
- Can be imported from/exported to CSV from Excel
- CSV is easy to read (when viewed in Excel)
- CSV is a defined standard (with a lot of variations regarding separator, TAB, Comma, Semicolon …)
  - … and it just works in Excel 😐

Location	Temperature	Time
Aarhus	22	2023-05-12 15.00.23
Viborg	25	2023-05-12 14.10.54

```
Location,Temperature,Time
Aarhus,22,2023-05-12 15.00.23
Viborg,25,2023-05-12 14.10.54
```

Case

- We want to find out what makes people happy looking at:
  - (name), age, salary, housing, civil status
- The data is structured, and we want an easy way to get an overview of our data
  - Excel seems to be the right choice

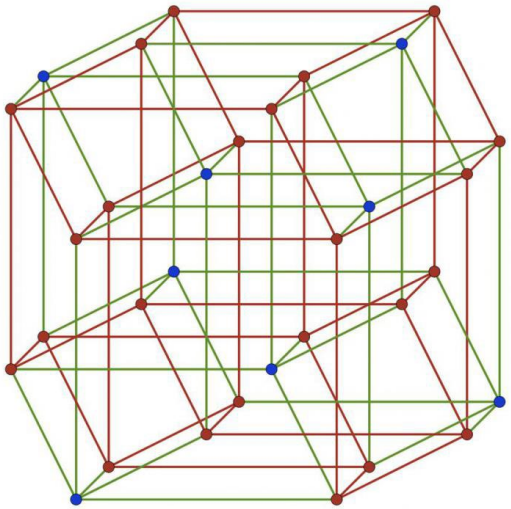
Name	Age	Salary	Housing	Civil Status
Jane	43	42.000	House	Relationship
John	34	31.000	Apartment	Married

Job done ☺

- … wait! “I think that people's children have an impact on their happiness”
  - We should add their **name** and **age**

Adding more than 2 dimensions

- Possible solutions
  - Add multiple values in the same cell and create a custom rule for how to separate them
  - Add repeated columns for each new sub-domain of data
  - Create a separate Excel-sheet for each domain and link them using some kind of id



Multiple values in same cell

- Hard to filter/count/sort on the individual values in the cell
- We now have created a custom format
  - Requires instructions for how to parse the values in the cell
- Adding more dimensions will make it even harder to work with
  - … we need to define a new rule for each added dimension

Name	Age	Children
Jane	43	**Luke/Jill**
John	34	**Joe**

Name	Age	Children
Jane	43	**Luke:20/Jill:15**
John	34	**Joe:9**

Repeat columns

- Hard to filter/count/sort on values in the cells as they span multiple columns
- We get a lot of empty cells
- Does not scale

Name	Age	Child1	Child2
Jane	43	**Luke**	**Jill**
John	34	**Joe**	

Name	Age	Child1-name	Child1-age	Child2-name	Child2-name
Jane	43	**Luke**	**20**	**Jill**	**15**
John	34	**Joe**	**9**		

New Excel sheet for each domain

- Hard to get and overview
- We have created a relational DB, but are missing the DBMS to query the data (make joins)

Persons

Id	Name	Age
1	Jane	43
2	John	34

Children

Id	Name	Age	ParentId
1	Luke	20	**1**
2	Jill	15	**1**
3	Joe	9	**2**

Alternative solutions

- XML
  - Text-based, ok to read
  - Verbose format
  - Hard to parse
    - requires extra information to determine types (xsd – Xml Schema Definition)
    - Different ways of nesting elements and defining attributes
- Relational DB
  - Binary, not directly readable
  - Relational tables connected with keys
  - Requires DB modelling skills
  - Requires a DBMS to create/read/update/delete data
- JSON
  - Text-based, easy read
  - Simple and concise format
  - Easy to parse

XML

```
<persons>
  <person name="Jane">
    <age>43</age>
    <salary>42000</salary>
    <housing>House</salary>
    <civilstatus>Relationship</civilstatus>
    <children>
      <child name="Luke">
        <age>20</age>
        <hobby name="Role Playing">
          <type>Acting</type>
        </hobby>
      </child>
       <child name="Jill">
        <age>15</age>
        <hobby name="Football">
          <type>Sport</type>
        </hobby>
      </child>
    </children>
  </person>
  <person name="John">
    <age>34</age>
    <salary>31000</salary>
    <housing>Apartment</salary>
    <civilstatus>Married</civilstatus>
    <children>
      <child name="Joe">
        <age>9</age>
        <hobby name="Football">
          <type>Sport</type>
        </hobby>
      </child>
    </children>
   </person>
</persons>
```

Relational DB (normalized)

housing

id	name
1	House
2	Apartment

persons

id	name	age	salary	housing_id	c_status_id
1	Jane	43	42000	1	2
2	John	34	31000	2	3

civil_status

id	name
1	Single
2	Relationship
3	Married

person_child_rel

person_id	child_id
1	1
1	2
2	3

hobby

id	name	hobby_type_id
1	Role Playing	1
2	Football	2

children

id	name	age	hobby_id
1	Luke	20	1
2	Jill	15	2
3	Joe	9	2

hobby_type

id	name
1	Acting
2	Sport

JSON

```
[
  {
    "name": "Jane",
    "age": 43,
    "salary": 43000,
    "housing": "House",
    "civilStatus": "Relationship",
    "children": [
      {
        "name": "Luke",
        "age": 20,
        "hobby": {
          "name": "Role Playing",
          "type": "Acting"
        }
      },
      { "name": "Jill", "age": 15, "hobby": { "name": "Football", "type": "Sport" } }
    ]
  },
  {
    "name": "John",
    "age": 34,
    "salary": 31000,
    "housing": "Apartment",
    "civilStatus": "Married",
    "children": [
      { "name": "Joe", "age": 9, "hobby": { "name": "Football", "type": "Sport" } }
    ]
  }
]
```

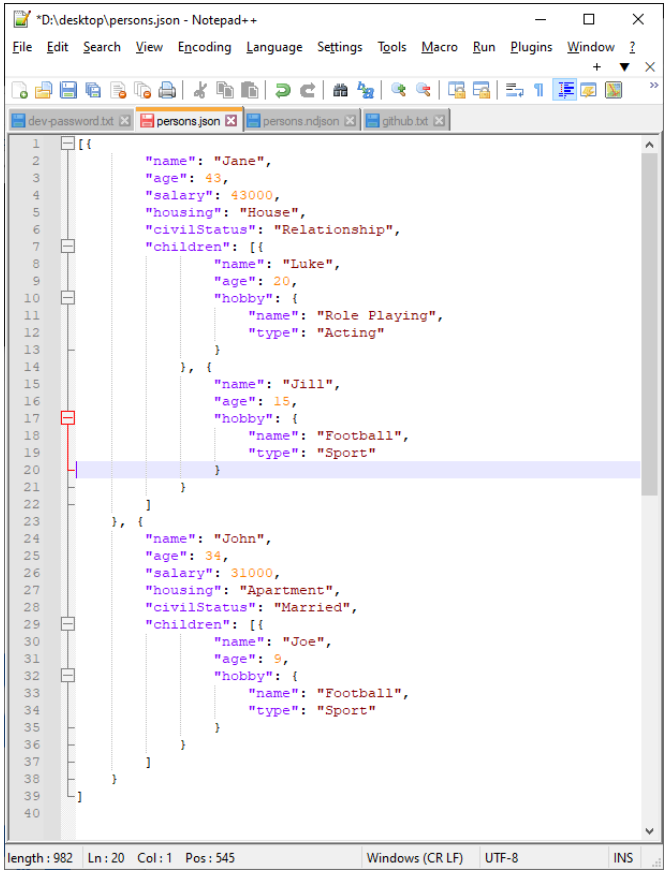
JSON structure and types

- Can start with an object or an array
- Objects and arrays can be nested indefinitely
- Property names must be surrounded by ""
- Arrays can contain any type. Each value must be separated by comma

```
{
  "text": "text content",
  "number": 2.2,
  "boolean": true | false,
  "object": {},
  "array": []
  "null": null
}
```

Creating and editing JSON

- It is just text, so any standard text-editor
  - Notepad
  - TextEdit
  - NOT Word as it should be edited as plain text
- For more support
  - Any code editor such as VSCode, IntelliJ products, Sublime Text…
  - JSON Viewer
  - Notepad++
    - Live code, simple list of books with 1-N genres



CHC
CENTER FOR HUMANITIES COMPUTING

Querying/filtering JSON

- Every programming language has JSON support but of cause requires programming skills

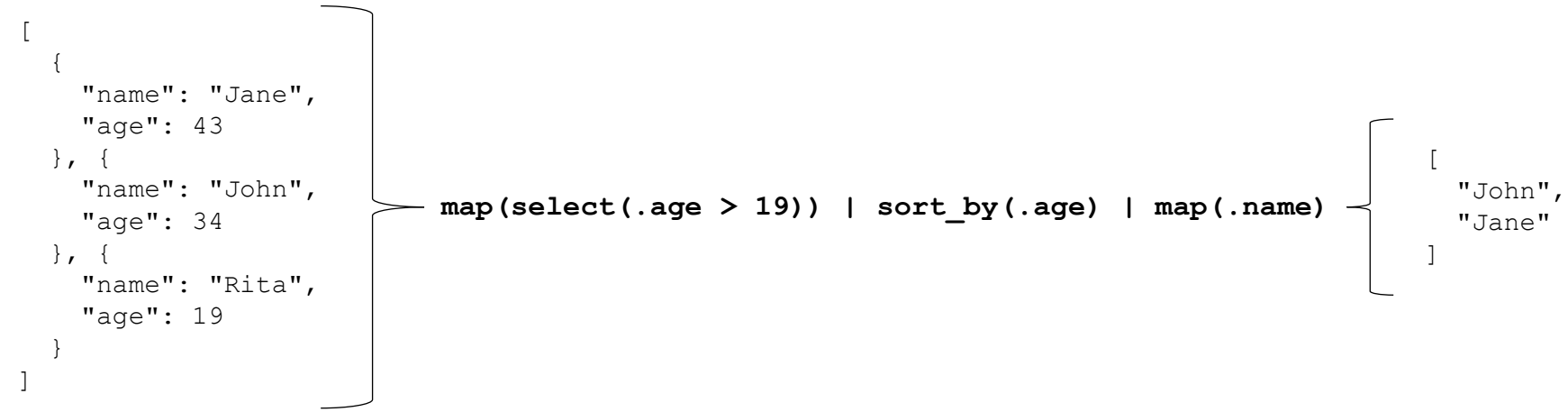
```
let persons = JSON.parse(fs.readFileSync('persons.json', 'utf8'));

let personCount = persons.length;
let avgSalary = persons.reduce((salary, p) => salary + p.salary, 0) / personCount;
let childrenOver10 = persons.flatMap(p => p.children).filter(c => c.age > 10).length;
```

- The command line program JQ is a strong alternative
  - Requires a little getting used to the commands (like formulas in Excel)
  - Very powerful for querying, filtering and transforming JSON

JQ

- Command line program
- Applies filters to a JSON input and outputs the transformed result
- Filters can be piped to other filters making complex filtering/transformations possible



CHC
CENTER FOR HUMANITIES COMPUTING

Jq Filters/functions

`.`	the current value (identity filter)
`.PROPERTY`	the property of the current value
`map()`	transform the input array to a new array with the result of the body of the filter
`select()`	return the input if the body evaluates to true otherwise no result
`contains()`	returns true if the input is contained by the body
`length`	The length of the input
`add`	Add all the elements of an array (x + y + z …)
`sort / sort_by()`	Sort an array of simple values or by object property
`|`	Pipe the result of one filter to a new filter
`Operators: +, -, * , / , <, >, ==, <=, >= … AND MORE`	Standard math and logical comparison, some are overloaded e.g. +
`… and many more filters…`	https://stedolan.github.io/jq/manual/

Install / try jq

- Installation from https://stedolan.github.io/jq/download/
- Online playground
  - https://jqplay.org/
  - https://jqkungfu.com/ (faster as it runs locally using web-assembly)
- Manuals
  - Manual: https://stedolan.github.io/jq/manual/
  - Cheatsheet: https://gist.github.com/olih/f7437fb6962fb3ee9fe95bda8d2c8fa4

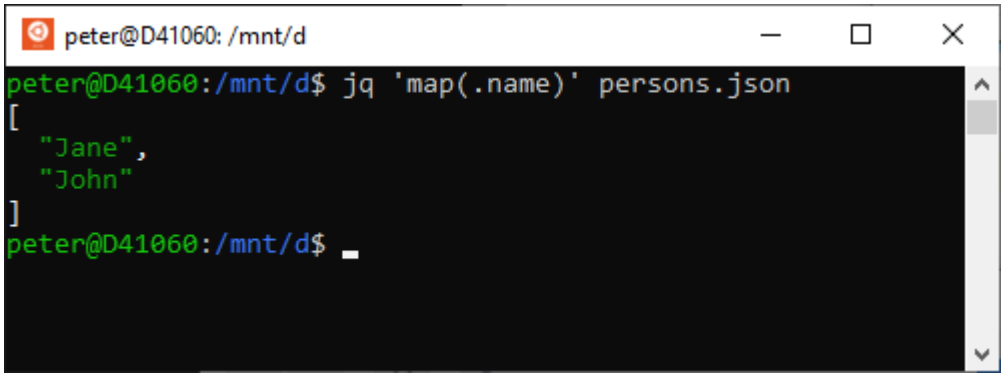
JQ examples

- Get the length of the persons-array
- Names sorted
- Persons with a salary > 40000
- Average salary
- Children over 10
- Persons with one or more children with a specific hobby

Execution of examples

- The following examples will only have the command / filter needed to get the desired result
  - If running the examples in e.g. https://jqplay.org/ this is what should be entered in the "filter" section
  - To run the examples using the command line program do the following (remember to put the command in single quotes)
  - PROMT> **jq** 'command/filter here' **path-to-file**

```
jq 'map(.name)' persons.json
```



Live programming

- We will try to run some of the tasks in https://jqplay.org/

```
[
  {
    "name": "Jane",
    "age": 43,
    "salary": 43.000,
    "housing": "House",
    "civilStatus": "Relationship",
    "children": [
      {
        "name": "Luke",
        "age": 20,
        "hobby": {
          "name": "Role Playing",
          "type": "Acting"
        }
      },
      { "name": "Jill", "age": 15, "hobby": { "name": "Football", "type": "Sport" } }
    ]
  },
  {
    "name": "John",
    "age": 34,
    "salary": 31.000,
    "housing": "Apartment",
    "civilStatus": "Married",
    "children": [
      { "name": "Joe", "age": 9, "hobby": { "name": "Football", "type": "Sport" } }
    ]
  }
]
```

Get the length of the persons-array

- The length filter outputs the length of an array

```
length
```

Select names and sort (asc/desc)

- Select name from each entry (`map(.name)`) and pipe (`|`) the result to a new query
- Sort the result (`sort`) … and pipe (`|`) the result to a new query
- … reverse the result (`reverse`)

```
map(.name) | sort
map(.name) | sort | reverse

OR …

[.[].name] | sort
```

Persons with salary > 40000

- Select all persons with a salary > 40000 (`map(select(.salary > 40000))`)
- … pipe (`|`) the result to a new query and select only name and age (`map({name, age})`)

```
map(select(.salary > 40000))

… only show name and age
map(select(.salary > 40000)) | map({name, age})
```

Average salary

- Select the salary of all persons (`map(.salary)`) and pipe (`|`) the result to a new query
- Add all numbers in the array (`add`) and divide them by the length of the array (`/ length`)

```
map(.salary) | add / length
```

Number of children older than 10

- Select all children (`map(.children[])`) and pipe (`|`) the result to a new query
- Select every child with age > 10 (`map(select(.age > 10))`) and pipe (`|`) the result to a new query
- Get the length (`length`)

```
map(.children[]) | map(select(.age > 10)) | length
```

Persons with child with hobby = "Role Playing"

- Select all person where condition is true (`map(select(…))`).
  - Select works like a filter where only the input where the filter is true is selected
- Condition is:
  - take each children array (`.children`) and pipe to new query
  - Take each child and produce `true` if "Role Playing" otherwise `false` (`map(.hobby.name == "Role Playing")`) and pipe to new query
  - If any of the booleans in the array are `true`, the result is true (`any`)

```
map(select(.children | map(.hobby.name == "Role Playing") | any))
```

Newline Delimited JSON

- Large datasets can be stored as newline delimited JSON where each line is a self-contained JSON-object/array
  - Makes it possible to read line-by-line
  - Store terabytes of data / millions of entries

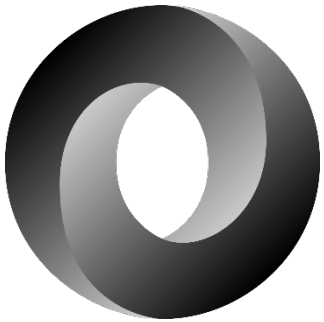
```
{"title": "Lord of the Rings", "author": "J. R. R. Tolkien", "genres": ["Fantasy", "Fiction"]}
{"title": "The Old Man and the Sea", "author": "Ernest Hemmingway", "genres": ["Fiction"]}
```

JQ and ndjson

- Use the `-s` flag to combine all objects into a single array

```
jq -s "map(.name)" persons.ndjson
```

Who uses JSON?



- Everybody…
- Almost all API's uses JSON as the standard data format
  - Twitter, Facebook, Google, NY-Times, InfoMedia, Danish Royal Library etc…
- Most open datasets
- CHC uses JSON as the standard format for storing and exchanging data
  - Geo-data
  - Twitter-, facebook-, reddit-data
  - web-site extracted data
  - meta-data for digitized documents
  - Some datasets are multiple terabytes ndjson with > 100.000.000 entries…

Workshop

- How are you structuring / storing / sharing data on your research projects?
- Does it work?
  - Why?
  - Why Not?
- How do you exchange data with other researchers?
  - Are there any obstacles/difficulties?
- Write below tabular data as JSON
  - Use https://jqplay.org/
  - Try to write a jq-selection for the names of the animals (in the filter section)
  - Try to run: `map(select(.continents | contains(["Asia"])))` to get all animals in Asia

name	speed	max age	continents
Jaguar	80	16	North America, South America
Elephant	40	70	Africa, Asia
Moose	55	20	North America, Europe, Asia

When to use a Relational DB?

- When data is updated regularly, and we must make sure redundant data is not present
- Many users needs to access the data at the same time

- Disadvantages
  - Requires a DBMS to access the data
  - Requires knowledge of the SQL-language
  - Requires knowledge about db modelling (normalization) as the schema is not easy to change after creation
- To solve the above programming/configuration of an application with a UI is typically required, but this takes time/cost money
  - CHC can help ☺