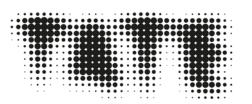
# Which data?













### What is data Refining, Wrangling, Engineering...

#### **WHAT**

- Removing duplicates
- Typos
- Joining field
- Data in wrong field
- Change of format (date)
- Encoding errors
- Join data sets
- Transpose row/column
- Enrichment

#### **AIMS**

- Integrating data
- Use & Re-use
- Sharing
- Analytics













**CSVKit** 





tripleGEO

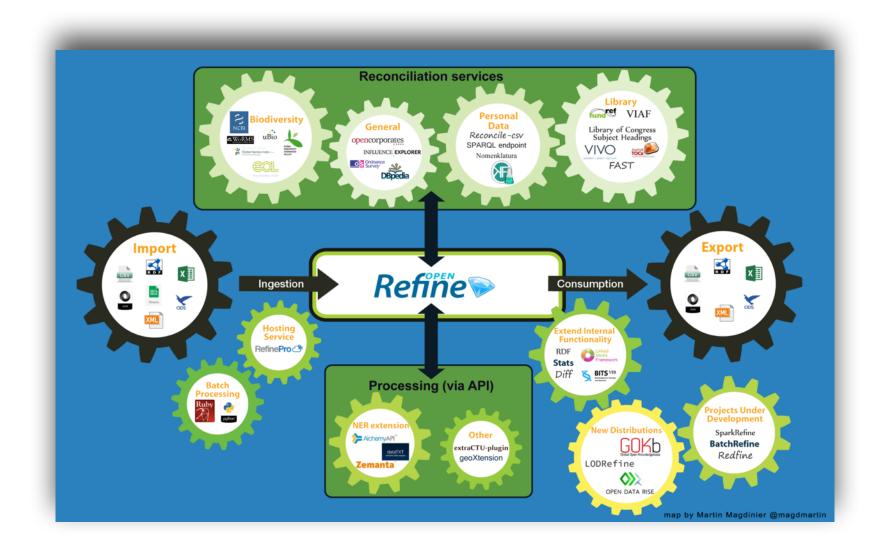


# **OpenRefine**

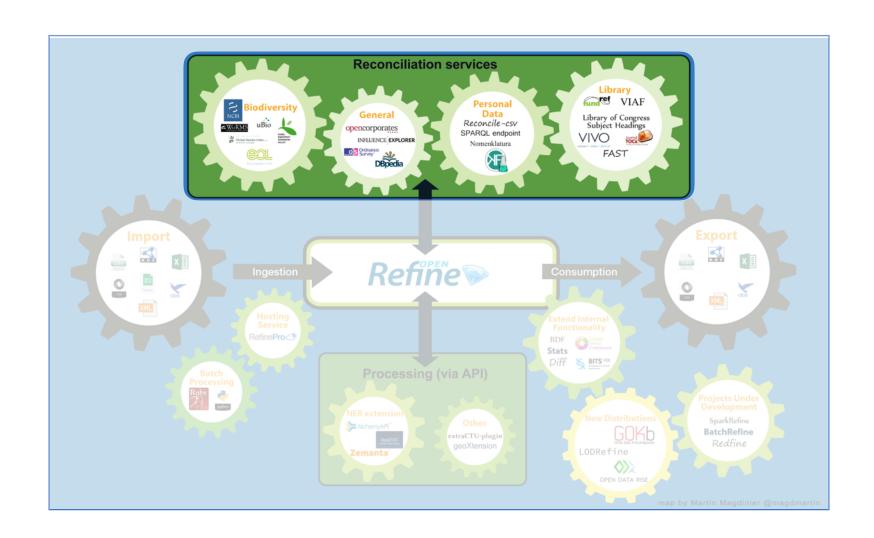


- Created by MetaWeb Technology as Gridworks in 2009
- Acquired by Google and rebranded as Google Refine
- Google continue its development: version 2.0 and 2.5
- Stop supporting in 2012
- OpenRefine is born: 2013

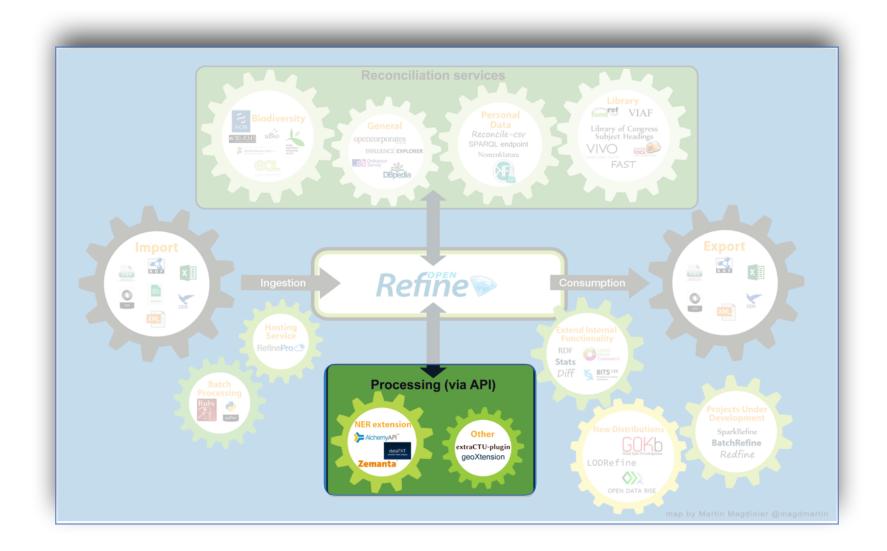




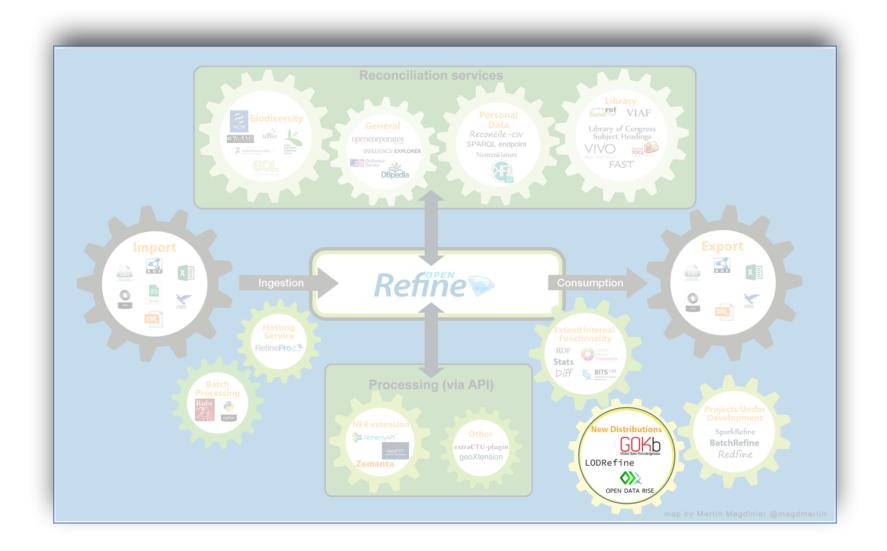












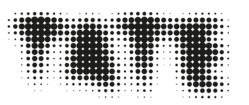


#### **Data: where?**

- Datahub <a href="http://datahub.io">http://datahub.io</a>
- Europeana <a href="http://www.europeana.eu">http://www.europeana.eu</a>
- Github <a href="https://github.com">https://github.com</a>
- EU OpenData portal <a href="https://www.europeandataportal.eu">https://www.europeandataportal.eu</a>
- Scraping





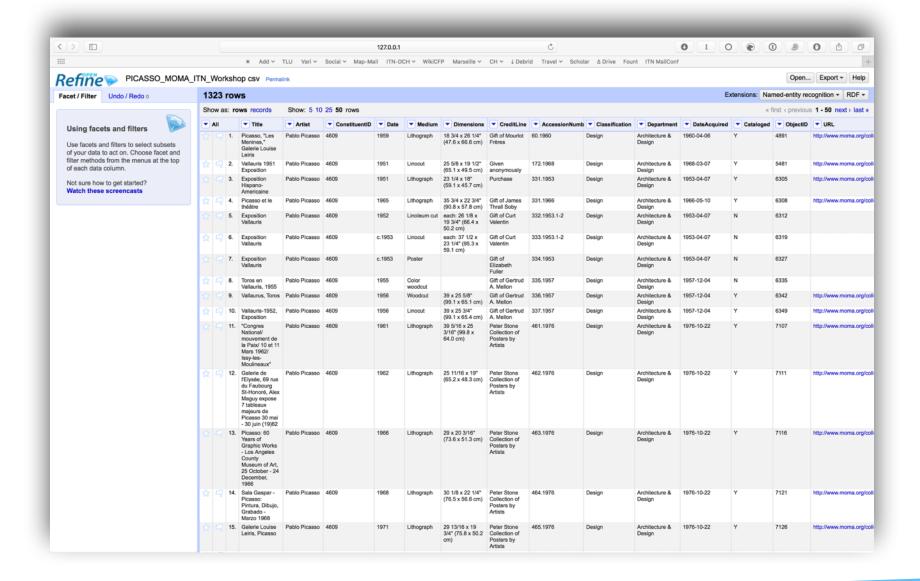




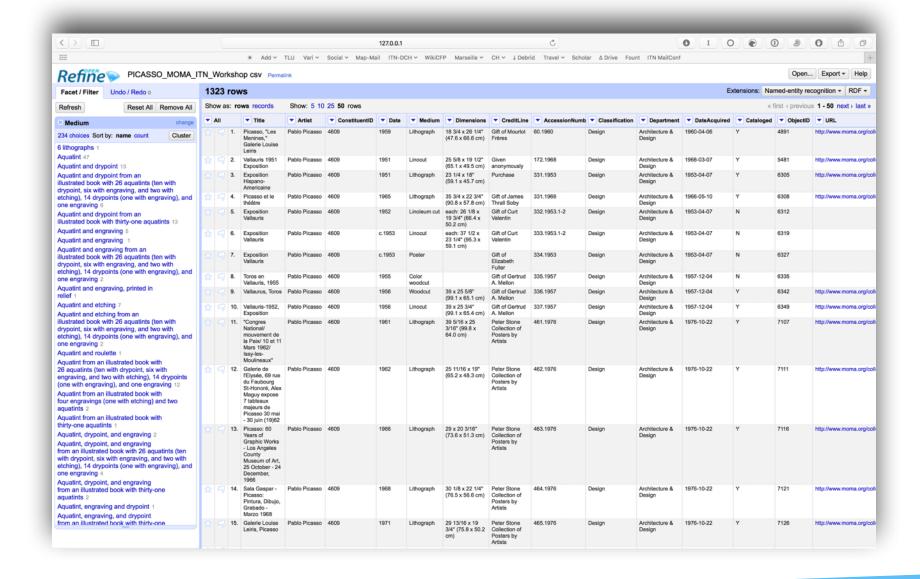












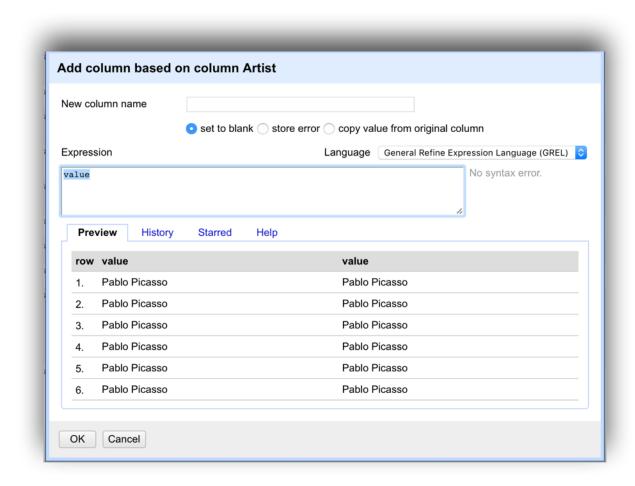


### **Medium & CreditLine**

## Faceting

- Key collision based algorithms
  - Fingerprint most conservative of all of them
  - ngram fingerprint more flexible, diversity in type of duplicate
  - metaphone 3 cologne-phonetic. Better check it.
- Nearest neighbour based algorithm
  - They calculate the number of edits between two strings and group them if under a certain threshold. Slower and require more computing power.







# **DateAcquired**

Change date format

toString(toDate(value),"dd/MM/yyyy")



# Date 1/4 - Jython

Date - Add column based on this column—> "start\_date"

```
import re
pattern = re.compile(r"\d{4}")
return pattern.findall(value)[0]
```

Date - Add column based on this column—> "end\_date"

```
import re
pattern = re.compile(r"\d{4}|(?<=-)\d{2}")
return pattern.findall(value)[-1]</pre>
```



# Date 2/4 - GREL

end\_date -> add "19" at the beginning of the date

return "19" + value

"start\_date" & "end\_date"

value.toDate()

end\_date - "Add column based on this column—> "diff"

diff(cells["end\_date"].value,cells["start\_date
"].value, "years")



# Date 3/4 - GREL

Add column based on this column—> "start\_date2"

value

start\_date - "Transform"

cells["end\_date"].value

end\_date - "Transform"

cells["start\_date2"].value



# Date 4/4 - GREL

date to string yyyy

toString(toDate(value), "yyyy")

Date -> remove column



#### **Cell Cross**

```
cell.cross("Picasso_place_ITN_Workshop", "year")
[0].cells["City"].value
```



#### Reconciliation

Compare value in my dataset with value from an external source. If they match link them and extract information









### Wikidata



- Wikimedia knowledge base
- Single source of structured information for wiki\*
- Organised with unique ID and attribute-value pair <attribute name, value>

#### **Reconciliation Wikidata**

https://tools.wmflabs.org/openrefine-wikidata/en/api

P569: Date of birth

P570: Date of death

```
'https://tools.wmflabs.org/openrefine-wikidata/
en/fetch_values?item=' + value +
'&prop=P569&label=true'
```



# **Adding information from Wikidata**

P569: Date of birth

```
'https://tools.wmflabs.org/openrefine-wikidata/
en/fetch_values?item=' + value +
'&prop=P569&label=true'

value.parseJson().values

value.replace('[','').replace(']','')

toString(toDate(value),"dd/MM/yyyy")
```



## **VIAF**



- International Authority File from OCLC
- Link diverse national authority file to a single virtual one
- A VIAF record gives access to all the national records

#### Reconciliation VIAF

http://localhost:8080/reconcile/viaf

Reconciliation with just one source

http://localhost:8080/reconcile/viaf/JPG (ULAN)

Retrieve IDs of a specific source

http://localhost:8080/reconcile/viafproxy/LC

cell.recon.match.id



#### **GEONAMES**



- Geographical database (WGS84)
- >10,000,000 geographical names
- Stable URI and accessible free of charge



#### Reconciliation Geonames

http://localhost:5000/reconcile

**URI** 

cell.recon.match.id

Coordinates

```
replace(substring(cell.recon.match.name,
indexOf(cell.recon.match.name, ' | ')), ' | ', '')
```

Name + Coordinates

cell.recon.match.name



# Add museum reference 1/3

Reconcile

Extract ID

cell.recon.match.name

Find the address

```
'https://tools.wmflabs.org/openrefine-
wikidata/en/fetch_values?item=' + value +
'&prop=P969&label=true'
```



## Add museum reference 2/3

### Replace characters

```
value.replace('[','').replace(']','').replace('"'
,'')
```

#### Check the API

```
"http://maps.google.com/maps/api/geocode/json?
sensor=false&address=" + escape(value, "url")
```



# Add museum reference 3/3

Create a new column

```
with(value.parseJson().results[0].geometry.locati
on, pair, pair.lat +", " + pair.lng)
```



## **OR** Add museum reference 1/1

#### Check the API

```
"http://maps.google.com/maps/api/geocode/json?
sensor=false&address=" + escape(value, "url")
```

#### Create a new column

```
with(value.parseJson().results[0].geometry.locati
on, pair, pair.lat +", " + pair.lng)
```



#### **Art and Architecture thesaurus**



Medium - Add column by fetching URL—> "AAT"

```
"http://leduc.gamsau.archi.fr/Skosmos/rest/v1/
aat/search?query=" + value + "*&lang=en"
```

Parse JSON -> Transform + Concatenation

value.parseJson().results

```
forEach(value.parseJson(),v,
[v.uri,v.prefLabel].join('||')).join('::')
```



### **Art and Architecture thesaurus**



Check first value

value.split("::")[0]

Split column



# **Exporting in XML?**

CSV to XML

https://shancarter.github.io/mr-data-converter/



## **Other Datasets**







