Data Transformation and Enrichment with OpenRefine: Part 2: Enhancing data

Ryan Mendenhall

Columbia University Libraries

October 10, 2018

Outline for the Workshop

Part 1:

Intros / About OpenRefine / Setting Up the Workspace / Review of key concepts from last week's workshop

Part 2:

About Reconciliation / Reconciliation and other enhancement techniques

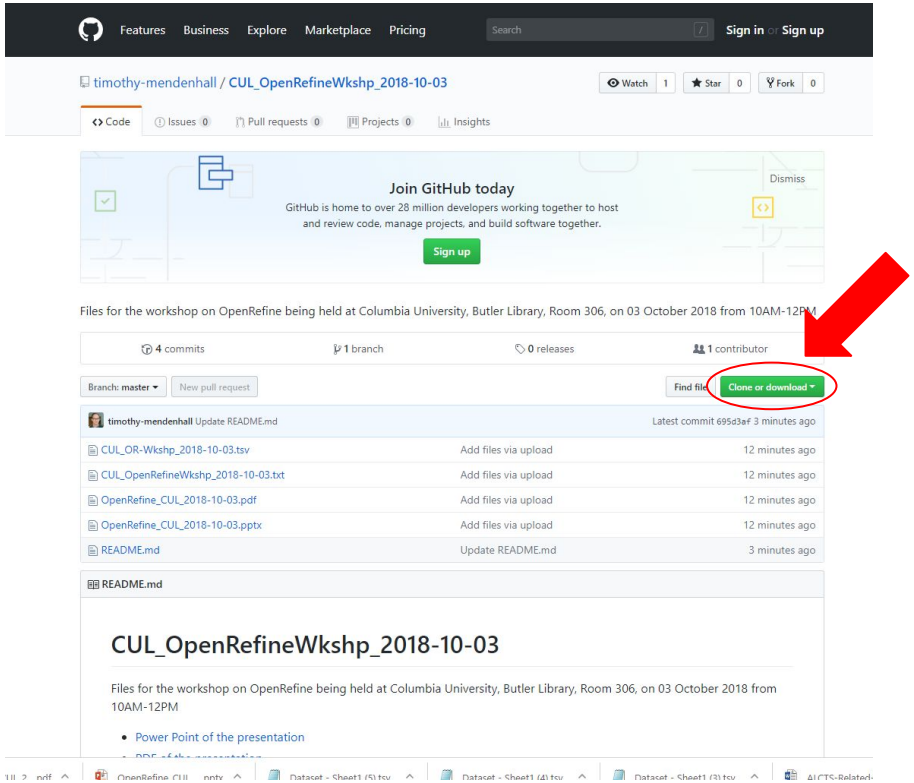
Introductions

- Name
- CUL Location / job
- Comfort level with linked data
- Working on a project that could incorporate external data?

Today's Workspace

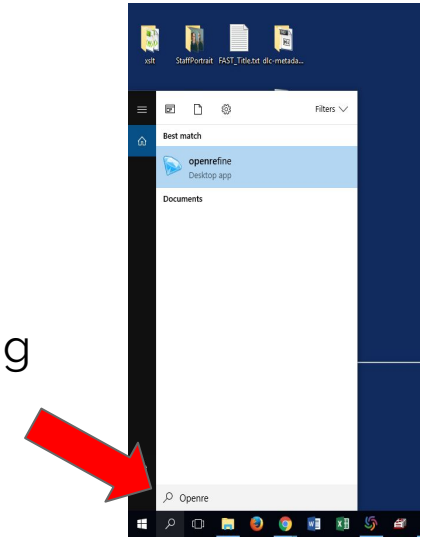
Download the workshop files
https://bit.ly/2C5FybG

Right click: Winzip → Extract all



Import data and create project

- Start the OpenRefine program
- If the browser window doesn't open: http://127.0.0.1:3333
- Import the dataset file and create a project (encoding Unicode, Tabs as delimiters, no quotes) using: **CUL_OpenRefineWkshp_2018-10-10.tsv**
- We'll use a second dataset for an exercise:
  - Import and create a project called CUL_DLC_Locations using
  - **CUL_DLC_Locations.tsv**



Scope of exercises

- Dataset *adapted* from an import spreadsheet created for RBML's Hubert H. Harrison Papers to upload content into CUL's Digital Library Collections (DLC)
- 4 methods of "reconciling" data: reconciliation service, API call, SPARQL endpoint, "crossing" with a second dataset

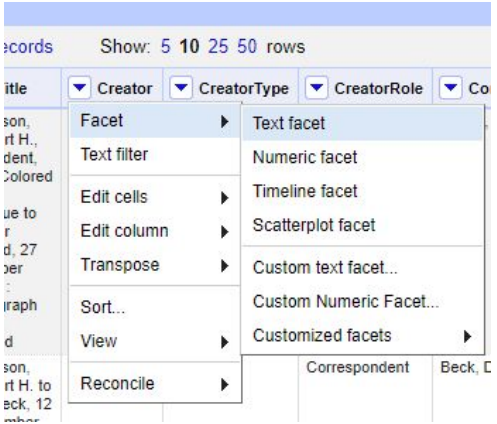
Review: What is OpenRefine?

- History: Freebase Gridworks → GoogleRefine → OpenRefine
- Currently on version 2.8
- Runs in the browser at http://127.0.0.1:3333
- GREL: Google Refine Expression Language
- Formats: delimited text (CSV, TSV), Excel, XML, JSON, RDF, web services / APIs / endpoints
- Documentation: GitHub, Website, email list, books (see resource list)

Review: Faceting

Facets are like filters:

If your data has facets applied, OpenRefine ***will ignore*** any rows not included in the facets when reconciling, exporting, or performing any other transformations.



Review: Rows and records

If you have “split” multivalued cells, keep in mind the following when faceting:

In record view, any row with a matching value in a multi-valued cell is included in a facet, including all “sibling” values in a multi-valued cell

In row view, only the matching values, not their “parent” rows or “sibling” values in their shared cells, are included in a facet

Linked Data: the basics

Basis: RDF triples model, "subject predicate object [SPO]," ideally all of which are represented as URIs

- [S] Charles Dickens [P] is author of [O] Oliver Twist →
- http://id.loc.gov/authorities/names/n78087607 :: http://id.loc.gov/vocabulary/relators/aut :: http://id.loc.gov/authorities/names/nr98012622

Vocabularies: BIBFRAME, CIDOC-CRM, RDF serializations of standards like MODS (underlying CUL's Digital Library Collections)

Formats: RDF-XML, JSON-LD, N-Triples, Turtle, and others. But can also be represented in a spreadsheet

Linked Data: some examples

SNAC

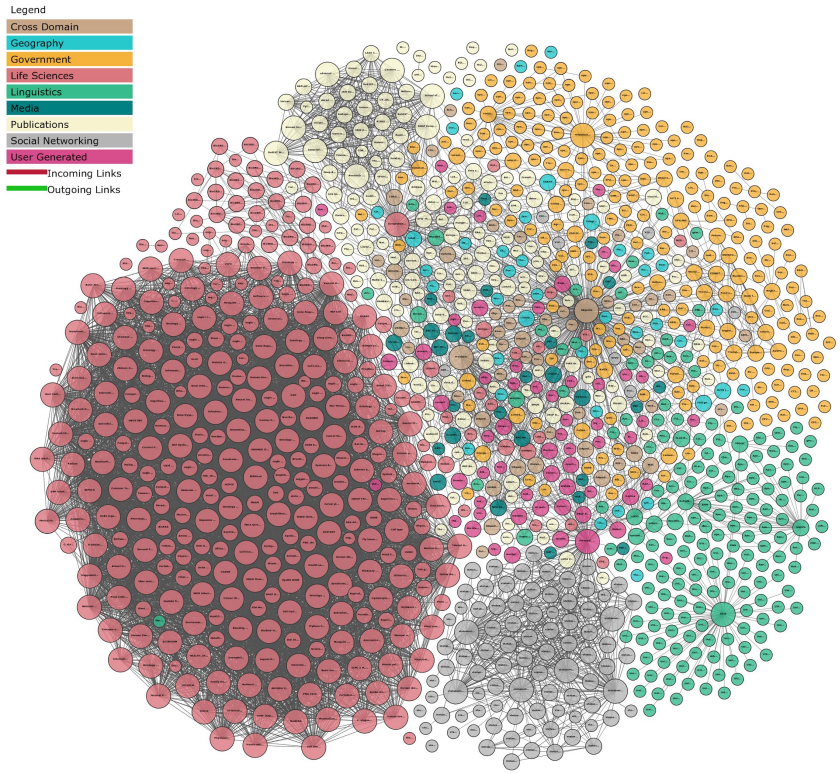
WorldCat Works and WorldCat Identities

Smithsonian American Art Museum online collections

Wikidata/Wikipedia/DBPedia

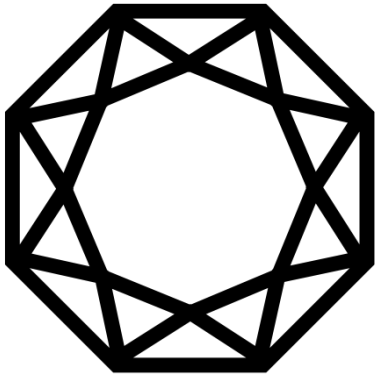
CUL: Hyacinth uses URIs, but doesn't expose linked data

Controlled Vocabularies



"Linking Open Data cloud diagram 2017, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/"

What is Reconciliation?



Created by Krizia Valtiare
from Noun Project

Reconciliation is a semi-automated process of matching text names to database IDs (keys).

https://github.com/OpenRefine/OpenRefine/wiki/Reconciliation

A reconciliation service is a web service that, given some text which is a name or label for something, and optionally some additional details, returns a ranked list of potential entities matching the criteria.

https://github.com/OpenRefine/OpenRefine/wiki/Reconciliation-Service-API

Reconciliation services

1. Reconciliation service queries a published linked data source
2. Reconciliation service performs fuzzy matching of our local data's names against the published vocabulary's names
3. Reconciliation service scores the returned matches and returns the top 3
4. In the returned results, you can access the following properties: id, name, type, and score
5. Reconciliation service simplifies quality checking of matches by providing a human-friendly interface to explore returned matches

OpenRefine 2.8 includes a Wikidata reconciliation service

conciliator

Reconciliation service by Jeff Chiu

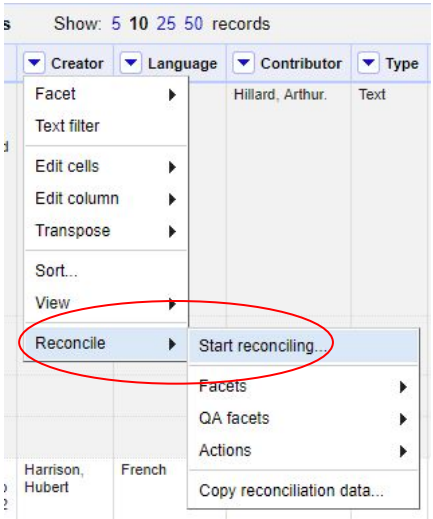
- http://refine.codefork.com

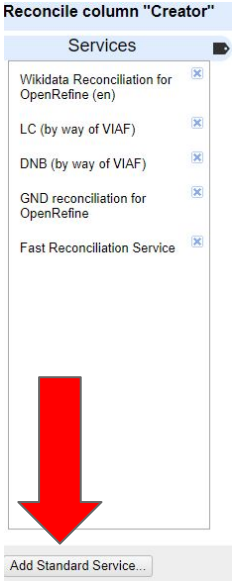
VIAF and LC/NAF by way of VIAF proxy

1. http://refine.codefork.com/reconcile/viaf
2. http://refine.codefork.com/reconcile/viafproxy/LC

Other possibilities: ORCiD, OpenLibrary, anything in VIAF via VIAF proxy

Adding a reconciliation service

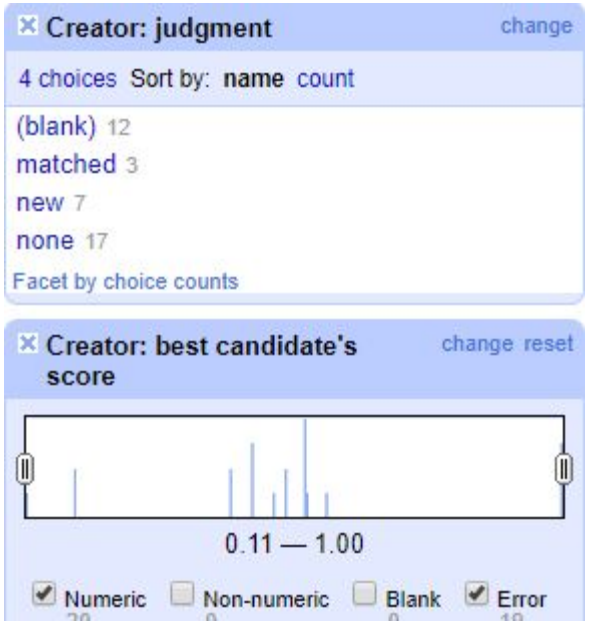




1. Reconcile → start reconciling
2. Add Standard Service
3. Enter URL of service
4. Click Add service
5. Working (it may take a little bit)
6. Your service should appear in the list if successfully added

Reconciliation facets

- **Matched**: the service found a near-exact match
- **None**: matches might (or might not) have been returned, but need review
- **New**: Manually flagged for further review

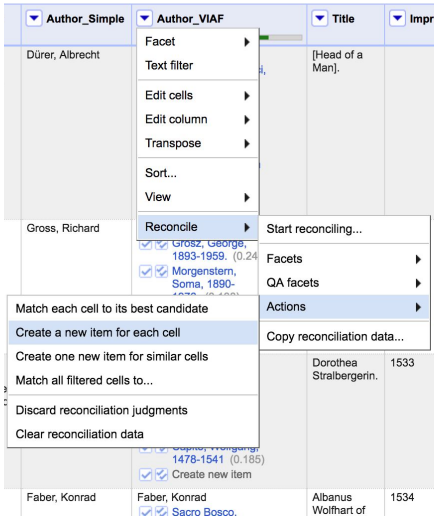


Reviewing matches

- Single check matches just that cell
- Double check matches all cells with that value
- As you approve matches via checkboxes, the rows are removed from the “None” facet

Other reconciliation actions

- Batch application of matches
- Creating new items: akin to starring / flagging -- adds cells to a “new” reconciliation facet to help filter for further review
- Discarding judgments: saves reconciliation data, but removes all matches
- Clear reconcilation data: wipes all reconciliation data



Exercise 1

1) Run the Conciliator reconciliation service LC/NAF by VIAF proxy on the Creator column. Use type “Person”
2) Review the facets--what has matched, what hasn’t matched?

GREL: extract URIs and names

- GREL: **cell.recon.match.id :** just grabs identifier (not always a full URI)
- **"http://id.loc.gov/authorities/names/"+cell.recon.match.id**
- GREL: **cell.recon.match.name** : grabs the preferred form or label of the reconciled term (some reconciliation services let you specify a preferred language and retrieve the preferred version of the term in that language)

Exercise 2

Using your reconciled Creator column:

Run Edit column→Add column based on this column, twice, using GREL to:

1. Extract the LC/NAF URI
2. Extract the preferred label from LC/NAF

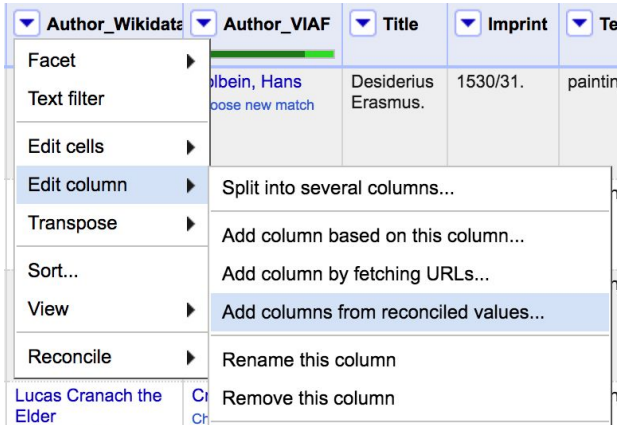
Don't worry--the GREL you need is in your cheatsheet

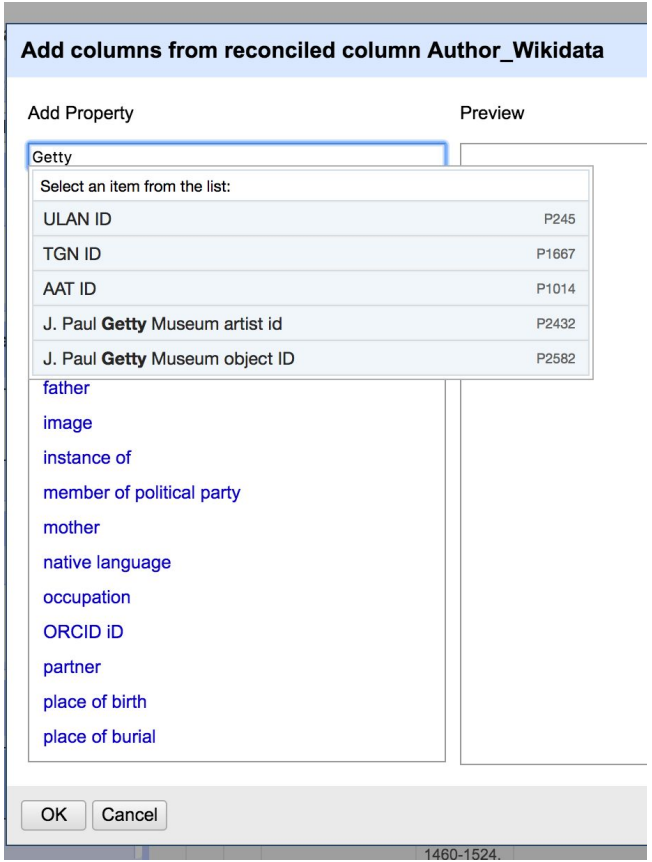
Wikidata reconciliation

OpenRefine comes “pre-loaded” with a very powerful Wikidata reconciliation service. The service has a couple nice added features that set it apart:

1. “Search for new match” enables you to edit values without re-running reconciliation
2. Built in “Data Extension API” that makes it easy to pull in other data from the Wikidata record

Wikidata: use Data Extension API





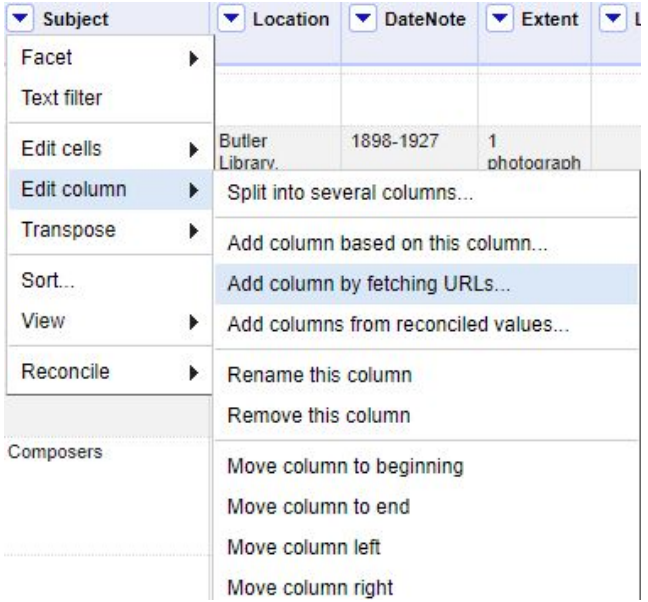
Exercise 3

Use the Creator column again

- Reconcile → Actions → Clear reconciliation data
- Run the Wikidata reconciliation service, use type Human
- Confirm some matches
- Once you've gotten some reconciled values:
  - Edit column→ Add column based on reconciled values
  - Add a column with another property from Wikidata!
  - Remember--OpenRefine will suggest a list of Wikidata properties, but you are not limited to this list

Querying APIs and other web data

Edit column→ Add column by fetching URLs



Querying APIs and other web data

Edit column→ Add column by fetching URLs

- This function enables you to pop your cell values into a URL, and then the results of calling the URL are returned in a new column
- Use this to "scrape" even unstructured data: no limit on what's returned--you can get the entire HTML code of a given page
- ***But*** most efficient for working with **APIs** and **endpoints**
- In the ensuing demos and exercises, don't worry about understanding the queries, the JSON, the SPARQL, and GREL--focus on understanding what's happening more broadly

Querying APIs and other web data

Disadvantages (versus using a custom reconciliation service)

- Slower (and some websites may lock you out)
- More error-prone
- More cleanup of reconciled data needed
- Better suited for “known-item” searching

Advantages

- Lightweight: no custom reconciliation service needed
- Flexible: can work with lots of kinds of online data sources
- Direct access to data unmediated by reconciliation
- Can get more than just identifier/URI and authorized heading

OCLC FAST API

- FAST: Faceted Application of Subject Terminology
  - Derived from LCSH to optimize online searching and navigating
  - Has 8 categories of terms--we'll look today at Topical
- API returns XML and JSON
  - GREL has functions to parse JSON

OCLC FAST API

- Sample query:
  - http://fast.oclc.org/searchfast/fastsuggest?query=VALUE&rows=30&queryReturn=suggestall+idroot+auth+cscore&suggest=autoSubject&queryIndex=suggest50&wt=json
- This query in GREL: See your “cheatsheet”
- Parsing out URI from returned results: Use GREL in your “cheatsheet”

Exercise 4

- Edit Cells → Split multivalued cells in Subject. Split on semicolon
- On Subject column: Edit column → Add column by fetching URLs
  - Insert GREL expression with FAST query from cheatsheet
- Switch to Rows view
- Use the second GREL expression on the cheatsheet to parse the URIs out of the returned JSON

Exercise 4: Notes

What happened? The FAST API is a bit problematic--it will return duplicate hits as well as some very unexpected results, chiefly because of how variant versions of a term are indexed.

The first GREL expression puts our cell value into an API call (a URL), replacing values that will gum up the FAST API

The second GREL expression looks at the returned results, compares them to our source column Subject, grabs the FAST identifier, de-dupes, and then reformats the FAST identifier to a URI

Getty Vocabularies: SPARQL Endpoint

http://vocab.getty.edu/queries

SPARQL = Sparql Protocol and RDF Query Language

```
SELECT ?x ?label
        {
?x skos:inScheme aat:;
        (xl:prefLabel|xl:altLabel)/gvp:term "painting"@en;
        skos:prefLabel ?label
        FILTER(lang(?label)="en").
        }
```

Exercise 5

Using the FormMezzanine column, run an Add column by fetching URLs task with the following query (see your cheatsheet):

'http://vocab.getty.edu/sparql.json?query=select+?x+?label{?x+skos:inScheme+aat:;(xl:prefLabel|xl:altLabel)/gvp:term"' + escape(value, 'url') + '"@en;skos:prefLabel+?label+filter(lang(?label)="en").}'

On the returned results, Edit Cells→ Transform with GREL:

forEach(value.parseJson().results.bindings,v,if(v.label.value == cells["Form"].value,v.x.value,"")).uniques().join("")

Exercise 5: Notes

What happened? Although Getty often specifies that a preferred version of a term have a qualifier, like “paintings (visual works),” including the qualifiers will invalidate your query. Without the qualifier, Getty’s SPARQL endpoint will often return multiple results, but we are only interested in one.

Therefore, I pre-processed the data to have a FormMezzanine column without the qualifier, but our second GREL script looks at our source column Form, compares the results returned from the Getty API, and grabs the URI of the result that matches the qualified version of the term in the Form column

“Crossing” datasets

GREL’s “cross” function can help us in the following instances:

- Querying web sources can be slow
- Some online datasets may only, or alternately, be made available as “data dumps” rather than through APIS or endpoints
- Some local controlled vocabularies may be inaccessible to the web, e.g. locations

“Crossing” datasets

Today’s use case: CUL locations (CUL_DLC_Locations.tsv)

- Not all CUL locations are represented in LC/NAF
- Some CUL locations prefer a local label for web display over the LC/NAF label
- CUL locations: small dataset
- CUL locations: local controlled vocabulary in Hyacinth inaccessible to web queries

“Crossing” datasets

- The second dataset needs to be loaded into your OpenRefine instance
- The Cross function compares the value of a cell with the values in a specified column of the second dataset
- If a match is found in 2nd dataset, the Cross function can pull in other data from that row in the second dataset

Exercise 6

If you haven't already, create a project with the file CUL_DLC_Locations.tsv ; name the project CUL_DLC_Locations

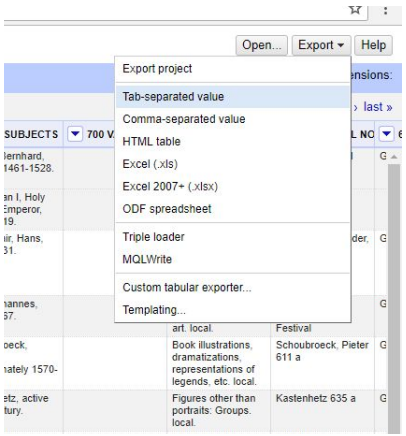
Using the the **Collection** column:

Edit column → Add column based on this column

GREL Expression (see your cheatsheet)

cell.cross('CUL_DLC_Locations','Label').cells['URI'].value[0]

Exporting your data



To get your data out of OpenRefine:

- Click Export (at the top right of screen)
- Pick a data format
- The file will appear in your default downloads folder

Exporting a project

- Export → Export project
- This exports both your data and the entire editing history since you loaded it into OpenRefine
- Can be opened by another instance of OpenRefine

Resources

- http://openrefine.org/
  - *Homepage has basic tutorials, Documentation has links to mailing lists, wikis, and Downloads has links to the GitHub page*
- van Hooland, Seth and Verborgh, Ruben. Linked data for libraries, archives and museums : how to clean, link and publish your metadata. London : Facet Publishing, 2014. 9781856049641
  - *A more in-depth exploration of the linked data world, with many examples of how to use OpenRefine*
- Stuart, David. Practical ontologies for information professionals. Chicago : Neal-Schuman, an imprint of the American Library Association, 2016. 9780838915110
  - *A very deep dive into linked data, the semantic web, and ontology development*

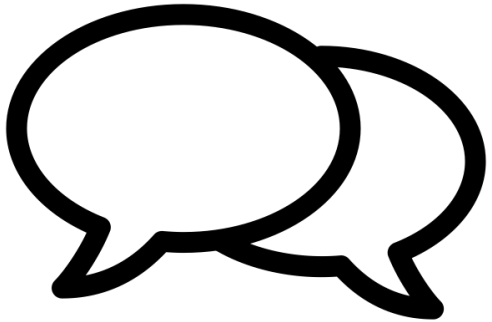
Resources ctd.

- Reconciliation services
  - Conciliator by Jeff Chiu: https://github.com/codeforkjeff/conciliator
  - Wikidata reconcile: https://tools.wmflabs.org/openrefine-wikidata/ (Installed by default in OpenRefine 2.8+)
  - Other reconciliation services:
    - fast-reconcile: https://github.com/timothy-mendenhall/fast-reconcile
    - lc-reconcile: https://github.com/cmh2166/lc-reconcile (last updated 3 years ago)
    - Getty AAT: https://github.com/ettorerizza/aat_reconcile (just released, still buggy)
- SPARQL
  - DuCharme, Bob. Learning SPARQL. Sebastopol, CA : O'Reilly Media, 2013. 9781449371432
  - Library Juice Academy: Certificate in RDF- and XML-based systems: the SPARQL fundamentals pt. 1-3. http://libraryjuiceacademy.com/certificate-xml-rdf.php

Resources ctd.

- Data sources
  - https://github.com/OpenRefine/OpenRefine/wiki/Reconcilable-Data-Sources
  - BARTOC (Basel Register of Thesauri, Ontologies, and Classification): https://bartoc.org/ : a list of possible data sources (e.g. controlled vocabularies)
  - *SPARQL endpoints*
    - W3C list of publicly available SPARQL endpoints (not exhaustive): https://www.w3.org/wiki/SparqlEndpoints
    - Additional endpoints not on W3C site:
      - The British Museum: http://collection.britishmuseum.org/sparql
      - Getty Vocabularies: http://vocab.getty.edu
      - CIA World Factbook (SNORQL and HTTP SPARQL endpoint): http://wifo5-04.informatik.uni-mannheim.de/factbook/
      - Linked Jazz: https://linkedjazz.org/sparql/
  - *OCLC API Explorer*: https://platform.worldcat.org/api-explorer/apis

Questions?



Ryan Mendenhall
trm2151@columbia.edu

Created by iconsmind.com
from Noun Project