Linked Data Web Application: use cases of UBC Open Collections API

Carolina Román Amigo

Supervisor: Richard Arias Hernandez, UBC SLAIS Instructor
Co-supervisor: Paul Joseph, UBC Systems Librarian

November 2017
# Table of Contents

**Introduction** 2  
Collections 2  
MacMillan Bloedel Limited fonds 2  
UBC Institute of Fisheries Field Records 3  
Vocabularies 4  
Geonames Ontology 4  
Encyclopedia of life 4  
Tools 5  
CARTO Builder 5  
OpenRefine 5  

**Implementation Process** 6  
MacMillan Bloedel Limited fonds 6  
Modelling 6  
Cleaning 7  
Reconciling 9  
Building interface 12  
UBC Institute of Fisheries Field Records 18  
Modelling 18  
Cleaning 19  
Reconciling 22  
Building interface 25  

**References** 29
Introduction

This project aims to explore the potential uses of the Open Collections Research API for linked data projects. The API is made available by University of British Columbia Libraries to provide machine-readable access to collections metadata and transcripts. We developed simple web applications providing data visualizations of collections metadata linked to external controlled vocabularies, providing an enhanced view of UBC’s digital repository. We describe below the collections, controlled vocabularies and tools used in this project.

Collections

MacMillan Bloedel Limited fonds
https://open.library.ubc.ca/collections/macmillan

The MacMillan Bloedel Limited fonds collection contains 2781 still images depicting the history of MacMillan Bloedel forest products company. Metadata contains names of locations (Library of Congress format) across the world, mainly in British Columbia. The goal for this use case is to link locations to a controlled vocabulary and plot records on a map, getting a preview of title, date and image thumbnail for each location.

Link for the demo:
https://carolamigo.carto.com/builder/881d9644-0439-4131-ad92-c16bcb1e2608/embed
The UBC Institute of Fisheries Field Records contains 11021 still images depicting pages of notebooks describing fish specimens collected around the world over a period of more than 100 years. Metadata contains latitude and longitude and species of fish collected for each record. The goal of this use case is to link fish species to a controlled vocabulary and plot records on a map, getting a preview of title and a link to species found in each location.

**Link for the demo:**
https://carolamigo.carto.com/builder/df748d01-2fc8-424b-ae46-dfd834b200a0/embed
Vocabularies

Geonames Ontology

http://www.geonames.org/ontology/documentation.html

Geonames is an ontology that provides geospatial semantic information to be used in the World Wide Web. It provides over 11 million geonames toponyms with unique URIs and an API service that can be used for reconciliation.

Encyclopedia of life

http://eol.org/

The goal of Encyclopedia of life is to provide access to knowledge about life on earth aggregating information scattered around the world in books, journals, databases, websites, specimen collections. It is the result of a collaborative initiative among academic institutions and the community. Although it is not a controlled vocabulary, it offers a unique URI to each record and a reconciliation service for OpenRefine.
Tools

CARTO Builder

https://carto.com/builder/

CARTO Builder is a web-based georeference visualization tool that allows you to easily build interfaces based on tabular data. By the time this report was written it was free for anyone to use, since you agreed to make your data publicly available. They also have free special licences for students (see more at GitHub Student Developer Pack).

OpenRefine

http://openrefine.org/

OpenRefine is a powerful tool to convert, clean and enrich data. Is if free to use and offers linked data extensions and reconciliation services to compare and combine related datasets.
Implementation Process

The implementation process for each use case is described in detail in this section, aiming to document and allow this project to be reproduced. The structure follows the workflow for Linked Data projects described by Hooland & Verborgh (2014).

MacMillan Bloedel Limited fonds

GitHub repository:
https://github.com/carolamigo/ubc_carto_macmillan

Modelling

- Download collection metadata using the Open Collections Research API. A php script to batch download is provided at OC API Documentation page > Download Collection Data. This script returns a folder containing one RDF file per collection item (or XML, JSON, any format preferred). We are going to use N-triples because the file is cleaner (no headers or footers), what makes the merging easier later. Edit the script following the instructions on the documentation page and run it using the command:

  $ php collection_downloader.php --cid macmillan --fmt ntriples

- Merge the files using the Unix cat command:

  $ cat * > merged_filename

- Convert merged file obtained to a tabular format. Import project in Open Refine using the RDF/N3 files option. No character encoding selection is needed.

Challenges

- My first try to get the collection metadata was to use Postman to send a POST query for the OC API using a key, as described in the OC API Documentation page. However, queries requesting more than 1000 items are not completed. Our collection is almost 3x larger than that limit, meaning that the data would have to be retrieved in batches and merged together in OpenRefine.
- OpenRefine is able to open N-triples files, but I had problems in the first try with predicates that are used more than once within the same record. For example, the predicate “subject”, used for keywords related to the resource, is repeated in several triples within a record. OpenRefine reads predicates as columns names (when using any RDF or triple based language decodification option), and it doesn't allow repeated columns names. In my second try everything worked perfectly, so I believe I had a small formatting problem in my previous file that was causing the problems with OpenRefine.

Cleaning

- Examine the metadata for Geographic Locations. With the tabular data open in OpenRefine, look for the column “http://purl.org/dc/terms/spatial”. In the column options, select “Facet” > “Text facet”. The facets show you all the unique values for geographic locations in this dataset. From that list, it is possible to see that:

  o The location names are following Library of Congress formatting style, with the province following the name of the city, and that they are in between double quotes with the language notation following:

    e.g. “Alberni (B.C.)”@en

  o Some location names have small typos:

    "Namaimo River (B.C.)”@en

  o Some resources have more than one geographic location associated with it:

    e.g. "Powell River (B.C.) ; Nanaimo (B.C)”@en

- Split the cells containing more than one geographic location value.

  o Duplicate the “http://purl.org/dc/terms/spatial” column using “Edit column” > “Add column based on this column” in order to preserve original values. Name the new column: spatial_cleaned

  o On the spatial_cleaned column, select “Edit cells” > “Split multi-valued cells”. The separator is “;”.


- Remove double quotes, provinces and "@en" from location names. Select “Edit cells” > “Transform” and write the following expression:

```javascript
value.replace("\"", " ").replace("@en"," ").replace(/\((\[\])\)+\)/, " ")
```

- Trim leading and trailing whitespaces by selecting “Edit cells” > “Common transforms” > “Trim leading and trailing whitespaces”.

- Cluster location names in order to combine entries with typos and small inconsistencies under just one geographic location name. On the spatial_cleaned column, select “Facet” > “Text facet”, then select “Cluster” in the facet window. In the cluster window, select Nearest neighbour” method. Select the “merge” box for “Nanaimo River”, correct the typo, and select “Merge selected and close”.

![Cluster & Edit column "spatial_cleaned"](image)

- Fill down column “subject”, “http://purl.org/dc/terms/title”, “http://purl.org/dc/terms/created” and “http://www.europeana.eu/schemas/edm/isShownAt“ as we have several orphan cells resulting from the triple to tabular data format conversion. Go to each column, “Edit cells” > “Fill down”.

8
Challenges

- Creating the right expression to manipulate data in OpenRefine can be challenging if you are not used to the GREL syntax and to regular expressions. The Regex Builder (https://regexr.com/) and the OpenRefine documentation on GitHub (https://github.com/OpenRefine/OpenRefine/wiki/Understanding-Expressions) are helpful resources.

Reconciling

- Configure Geonames reconciliation service in OpenRefine following the procedure described here: https://github.com/cmh2166/geonames-reconcile. This procedure involves getting a Geonames API username, installing python packages, cloning the code in the GitHub repository above, and running the script provided in the code.

- Perform reconciliation on the “spatial_cleaned” columns using “geonames/name_equals” option. Follow the steps described here: http://christinaharlow.com/walkthrough-of-geonames-recon-service.

- When reconciliation is finished, review the results. An easy way to do it is to facet by text (“Facet” > “Text facet”) and filter results by clicking on a location name on the facet menu. On the spatial_cleaned column, click on the link for the location to check the reconciled value. This will open the Geonames window with location information. If it is correct, no further action is required. If it is wrong (e.g. Alice Lake found is in B.C. but geonames returned a lake with the same name in Michigan), click on “Choose a new match” under any of the wrong entries on the spatial_cleaned column. Three options will show up. Select the correct one by using the double checked box, which means your decision will be applied to all other cells that match this condition. If no correct option show up in the cell, click on the double checked box of “Create new topic”, meaning that no reconciliation value will be added to cells that match this condition. There are 66 unique values in this dataset, so it is possible to review one by one until it is done.
• Verify reconciliation results that didn't find a match (including the ones you had to "create a new topic" for) by selecting the "none" facet in the judgment box. I have found the following ones with no matches, so I had to add coordinates manually for those by looking up manually in the Geonames database using the Geonames search box ([http://www.geonames.org/](http://www.geonames.org/)). To mass edit a value, click on the edit link that appear next to the exclude link for the value in the facet window. Enter the value and click "Apply".
- Extract reconciled data retrieving name, id and latitude/longitude, as a string separated by “|”. Select `spatial_cleaned`, “Edit cells” > “Transform”, and entering the following expression:

```plaintext
cell.recon.match.name + " | " + cell.recon.match.id
```

- Split the values obtained in the reconciliation, that should be in this format:

```
Nanaimo River | 49.1304, -123.89385 | http://sws.geonames.org/6951400
```

Select `spatial_cleaned`, “Edit column” > “Split into several columns”. Select the “|” separator and name the columns according after split: `geonames_names`, `geonames_coord`, `geonames_uri`.

- The “`geonames_coord`” column has to be further split in latitudes and longitudes using the same command above, “Edit column” > “Split into several columns”, with separator “,”. Name the columns “lat” and “long”. Trim leading and trailing whitespaces by selecting “Edit cells” > “Common transforms” > “Trim leading and trailing whitespaces”.

## Challenges

- Setting up the Geonames reconciliation service on OpenRefine was not straightforward, since I had to install more than one Python package to make it work.
- My first try to reconcile the data was to download the Geonames database dump for the countries that appear in the collection and constructing URIs based on each location id using OpenRefine. Then I used VLOOKUP function on Excel to reconcile the collections list of locations with the geonames spreadsheet, using the timezone info to check if the location retrieved is correct. The results were good, but this is a more laborious way that
Building interface

- Prepare the data to interface. In order to have links and images on CARTO interface, we have to add html tags in the source dataset.

  - Remove double quotes and language. Create a new column “title” based on the column “http://purl.org/dc/terms/title”, using the following expression:

    ```javascript
    value.replace("\"", " ").replace("@en"," ")
    ```

  - Add html tags for title links. Create a new column “title_link” based on the column “subject”, using the following expression:

    ```javascript
    "<a href=""+value+"">"+if(isBlank(cells["title"]), " ", cells["title"]).value+"</a>"
    ```

  - Remove double quotes and language. Create a new column “date” based on the column “http://purl.org/dc/terms/created”, using the following expression:

    ```javascript
    value.replace("\"", " ").replace("@en" ,"")
    ```

  - Add html tags for location links. Create a new column “geoname_link” based on the column “geonames_uri”, using the following expression:

    ```javascript
    "<a href=""+value+"">"+if(isBlank(cells["geonames_names"]), " ", cells["geonames_names"]).value+"</a>"
    ```

  - Add html tags and links for images. Create a new column “image_link” based on the column “http://www.europeana.eu/schemas/edm/isShownAt”, using the following expression:

    ```javascript
    "<img width="188"
    src="http://iiif.library.ubc.ca/image/cdm.macmillan."+value.substring(10,19).replace(".","-")+".0000" +"/full/150,0/default.jpg"/>>"
    ```
• Export the dataset from OpenRefine in .csv format. Name the file “mcmillan_cleaned”.

• Sign up or Log in to CARTO Builder: https://carto.com/signup/. Create a new map and import the Open Refine exported file to your map.

• Georeference your dataset following the instructions here: https://carto.com/learn/guides/analysis/georeference. Once in your map, click on the dataset on the left menu, then click on “Analysis” > “Add analysis” > Georeference. Select the corresponding column names in your dataset for latitude and longitude (lat and long). Note that the application is plotting just one resource per location, so we will need to aggregate the results to have all the resources plotted.

• Export the georeferenced dataset from CARTO in csv format, in order to incorporate the “the_geom_webmercator” column (with georeferenced data) in your dataset. Name the file “mcmillan_cleaned_geo”. Import the dataset back into CARTO map, and delete the previous dataset from your map. This step is necessary since CARTO does not allow georeference analysis and SQL manipulation (that we will need for aggregation) of the data concomitantly.

• Click on the dataset on the lateral menu and select the “Data” tab. At the bottom of the lateral panel, enable SQL option. Paste the following query in the editor and click “Apply”:

```sql
SELECT string_agg(DISTINCT CONCAT (date, ' <br>', title_link, ' <br>', image_link, ' <br><br> '), '<br><br><br><br>') as new_column_aggregated, geoname_link, the_geom_webmercator, Min(cartodb_id) cartodb_id
```

13
FROM mcmillan_cleaned_geo

group by geoname_link, the_geom_webmercator

- Click on the “Pop-up” tab, and enter the following settings:
  - Style: color
  - Window size: 240
  - Header color: #b6dc9
  - Show item: check all boxes

- Click on any point on the map, you should see something similar to this:

- To build the filters:
  - Click on “mcmillan_cleaned_geo” dataset, “style” tab, and use the following settings:
    - Aggregation: none
    - Size 10, color #d1c4c4
    - Stroke 1 color #FFFFFF
    - Blending: darken

  - Exit “mcmillan_cleaned_geo” dataset and click on the “add” button to re import the same dataset. You will get copy named “mcmillan_cleaned_geo_1”. Click on this new dataset, “style” tab, and use the following settings:
    - Aggregation: none
    - Size 15, color #ee4d5a
    - Stroke 1 color #FFFFFF
- Blending: none

  - Exit the dataset and make it the second one in the list showing on the lateral panel, dragging and dropping it. We want this new layer behind the one that has the pop-up with photos.

  - Click on “Widget” tab to add filters. Add widgets following the instructions here: [https://carto.com/learn/guides/widgets/exploring-widgets](https://carto.com/learn/guides/widgets/exploring-widgets).

- Publish your map using the “Publish” button on the left lateral panel, just under the map name. Final result:

![Map with widgets and filters](image)

**Challenges**

Solving the problem of showing more than one resource at the same location was tricky. My first try was to use some sort of cluster visualization. I built the map outside CARTO using CARTO.js following procedures here:

- [https://github.com/Leaflet/Leaflet.markercluster](https://github.com/Leaflet/Leaflet.markercluster)
- [http://bl.ocks.org/oriolbx/7518b5834d1b679759bda218871cb315](http://bl.ocks.org/oriolbx/7518b5834d1b679759bda218871cb315)

I used the following Carto.js URL to build app outside CARTO:
I was able to do it but we had locations with more than a 1000 records, so the interface was impossible to navigate:

My second try was to do it using CARTO data options. It allows data manipulation using SQL, but it took me some trial and error and help from the following resources to get it right:

- [https://gis.stackexchange.com/questions/135769/cartodb-displaying-multiple-items-at-same-address](https://gis.stackexchange.com/questions/135769/cartodb-displaying-multiple-items-at-same-address)
- [https://gis.stackexchange.com/questions/90578/cartodb-aggregate-and-infowindow](https://gis.stackexchange.com/questions/90578/cartodb-aggregate-and-infowindow)

This was the result I got with the right SQL query (using string_agg, CONCAT and Min):
My first try to get clickable links and images inside the pop-ups was to edit the CSS and HTML in CARTO “style” and “pop-up” tabs. I was not able to do it because the aggregate SQL statement was not allowing me to add more columns from the data being retrieved from the table. I was able to build the string with CONCAT, but when I looked at the table the columns concatenated were not there, just the resulting aggregated column. If I was able to retrieve the columns separately as well, it would be possible to edit the html for the pop-ups using moustache {} (https://mustache.github.io/).

So I changed my approach and added the html tags directly on the source code. In order to display images, IIIFs URLs had to be generated in Open refine using OC API instructions.

Finally, adding filters was tricky because of the same limitation I had with the use of the SQL query. The solution was to create a new layer with a duplicate of the dataset, without the aggregation query. This duplicated layer has all the original dataset columns accessible, and contains only filters, no pop-ups. Moving this layer behind the one with the pop-ups keeps them accessible.
Modelling

- Download collection metadata using the Open Collections Research API. A php script to batch download is provided at [OC API Documentation page](#) > Download Collection Data. This script returns a folder containing one RDF file per collection item (or XML, JSON, any format preferred). We are going to use N-triples because the file is cleaner (no headers or footers), what makes the merging easier later. Edit the script following the instructions on the documentation page and run it using the command:

```
$ php collection_downloader.php --cid fisheries --fmt ntriples
```

- Merge files using the following python script. The folder containing the files to be merged has to be named “fisheries” and the script has to be run from the same folder the folder fisheries is in:

```
#adapted from: https://stackoverflow.com/questions/17749058/combine-multiple-text-files-into-one-text-file-using-python

import glob
read_files = glob.glob("fisheries/*.txt")
with open("result.txt", "wb") as outfile:
    for f in read_files:
        with open(f, "rb") as infile:
            outfile.write(infile.read())
```

- Convert merged file obtained to a tabular format. Import project in Open Refine using the RDF/N3 files option. No character encoding selection is needed.

**Challenges**

*The Unix cat command is not suitable to merge a large number of files. I decided to use a python script to do it after getting stuck in an infinite loop by trying to use:*
Cleaning

- The latitude values are in the “http://www.w3.org/2003/01/geo/wgs84_pos#lat” column. We have to change their formatting to numbers so CARTO can understand them:

"54 13"@en > 54.13
"0 30 S"@en > -0.3 (Latitudes south of the equator have negative values)

Create a new column “test” based on the column “http://www.w3.org/2003/01/geo/wgs84_pos#lat” using the following expression to remove any character and preserve only digits and blank spaces (keeping the spaces is important for placing the decimal points later):

value.replace(/[^\ ,^\d]/, "")

We have now to transform those values in numbers, but it is important to insert the decimal point in the right spot first, so select on the column “test”, “Edit column” > “Split into several columns”, separating by a blank space and selecting splitting into 2 columns at most. You are going to get two columns, “test 1” and “test 2”. Create a new column “latitude” based on the “test 2” column using the following expression to concatenate the values with a decimal dot in between:

 cells["test 1"].value + "." + cells["test 2"].value

On “latitude” column, select “Edit cells” > “Transform” and write the following expression to remove any remaining blank spaces:

value.replace(" ","")

We have now the values with the decimal point in the right position. Ensure all values are numbers by selecting on the column “latitude” > “Edit cells” > “Common transforms” > “To number”. Delete columns “test 1” and “test 2”.
Filter column “http://www.w3.org/2003/01/geo/wgs84_pos#lat” to select only cells containing “S”, using “Text filter” and typing “S” in the box that appears in the left sidebar. On the “latitude” column, select “Edit cells” > “Transform” and write the following expression to make all south latitudes negative values:

\[
\text{value} \times -1
\]

Now we have all latitudes south with a negative sign before them. Close the “Text facet” window on the left sidebar to remove the filter.

- We have now to repeat the procedure to the “http://www.w3.org/2003/01/geo/wgs84_pos#long” column (longitudes).

Create a new column “test” based on the column “http://www.w3.org/2003/01/geo/wgs84_pos#long” using the following expression to remove any character and preserve only digits and blank spaces:

\[
\text{value}.replace(/[^\,\d]/, "")
\]

We have to transform those values in numbers, but it is important to insert the decimal point in the right spot, so select on the column “test”, “Edit column” > “Split into several columns”, separating by an blank space and selecting splitting into 2 columns at most. You are going to get two columns, “test 1” and “test 2”. Create a new column “longitude” based on the “test 2” column using the following expression to concatenate the values with a decimal dot in between:

\[
\text{cells["test 1"].value} + "." + \text{cells["test 2"].value}
\]

On “longitude” column, select “Edit cells” > “Transform” and write the following expression to remove any remaining blank spaces:

\[
\text{value}.replace(" ", ",")
\]

We have now the values with the decimal point in the right position. Ensure all values are numbers by selecting on the column “longitude” > “Edit cells” > “Common transforms” > “To number”. Delete columns “test 1” and “test 2”.

Filter column “http://www.w3.org/2003/01/geo/wgs84_pos#long” to select only cells containing “W”, using “Text filter” and typing “W” in the box that appears in the left sidebar. On the “longitude” column, select “Edit cells” > “Transform” and write the following expression to make all west longitudes negative values:

\[
\text{value} \times -1
\]
Now we have all longitudes west with a negative sign before them. Close the “Text facet” window on the left sidebar to remove the filter.

- Let’s verify if the values for latitude and longitude are within the correct ranges. Facet the “longitude” column by number (“Facet” > “Numeric facet”) to check values (you might need to increase the faceting limit number). Longitudes have a range of -180 to +180. Any value outside that range is incorrect. Slide the filter selector on the left sidebar to see values that are larger than 180. Uncheck box “blank”. Take a look on the rest of the metadata for inferring the correct value. Correct then manually by clicking on edit inside the wrong value cell, changing the value, the data type to “number” and selecting “Apply to all identical cells”.

Now facet the “latitude” column by number (“Facet” > “Numeric facet”) to check values. Latitudes have a range of -90 to +90. Any value outside that range is incorrect. Slide the filter selector on the left sidebar to see values that are larger than 90. Uncheck box “blank”. We can see by examining column “longitude” that these values of latitude and longitude are swapped. Correct then manually by clicking on edit inside the wrong value cell, changing the value, the data type to “number” and selecting “Apply to all identical cells”. Changes to latitude and longitude are complete.

- The fish species are in the “http://purl.org/dc/terms/subject” column. To get better reconciliation results, we have to remove the double quotes, the “@en”, the “sp.” and keep just the species name inside the square brackets (when it exists):

"Agonus acipenerinus [Agonus accipenserinus]"@en  >  Agonus acipenerinus
"Ambassis sp."@en  >  Ambassis

Create a new column “species” based on the column “http://purl.org/dc/terms/subject”, using the following expression:

```
value.split('[')[-1].replace("\", " ").replace("@en"," ").replace("sp.","").replace("]","'')
```

- Trim leading and trailing whitespaces by selecting “Edit cells” > “Common transforms” > “Trim leading and trailing whitespaces”.

- Cluster species names in order to combine entries with typos and small inconsistencies under just one species name. On the “species” column, select “Facet” > “Text facet”, then select “Cluster” in the facet window. In the cluster window, experiment with different clustering methods. Start with “key collision” > fingerprint. Take a look at the results, and, if they are good enough, select the “Select all” button and then “Merge
selected and Re-Cluster”. Iterate until there are no more cluster formed, then try another clustering method until your have formed all clusters possible.

- Fill down the following columns as we have several orphan cells resulting from the triple to tabular data format conversion. Go to each column, “Edit cells” > “Fill down”.
  - “subject”
  - “http://purl.org/dc/terms/title”
  - “http://purl.org/dc/elements/1.1/date”
  - “http://www.europeana.eu/schemas/edm/isShownAt”
  - “http://purl.org/dc/terms/coverage”
  - “http://purl.org/dc/terms/spatial”
  - latitude_number
  - longitude_number

**Challenges**

To clean latitude and longitude values was the hardest part. It took me some trial and error and facet playing to get to know the data well enough to clean it, because it was far from uniform.

Reconciling

- On the “species” column, select “Reconcile” > “Start Reconciling” > “Add standard service”. Paste the following URL* in the box, then click “Add Service”:

  http://iphylo.org/~rpage/phyloinformatics/services/reconciliation_eol.php
*The Encyclopedia of Life (EOL) taxonomy reconciliation service to Open Refine was developed by: http://iphylo.blogspot.ca/2012/02/using-google-refine-and-taxonomic.html

- The reconciliation service will appear under reconciliation services tab. Select it and click “Start reconciling”. The process will take a long time (one hour or two) since we have many entries. You have to wait until it is done to do any further work on the data.

- When reconciliation is finished, review the results. Use the reconciliation faceting “species: judgment” box on the left sidebar to review the “none” ones. Those need you input to pick the best match. Up to three options show up. Select the correct one by using the double checked box, which means your decision will be applied to all other cells that match this condition. If no correct option show up in the cell, click on the double checked box of “Create new topic”, meaning that no reconciliation value will be added to cells that match this condition (they are going under “new” facet and you will need to add values manually for those later).
As there are too many unique values to assess, you can review a sample and then, with the “none” facet still on, select on the species column “Reconcile” > “Actions” > Match each cell to its best candidate.

- Extract reconciled data retrieving name and id, as a string separated by “|”. Select “species”, “Edit cells” > “Transform”, and entering the following expression:

\[
\text{cell.recon.match.name} + "|" + \text{cell.recon.match.id}
\]

- Split the values obtained in the reconciliation, that should be in this format:

\[
\text{Isopsetta isolepis (Lockington, 1880)} | 995111
\]
Select “species”, “Edit column” > “Split into several columns”. Select the “|” separator and name the columns according after split: species_eol, eol_id.

- We have to build EOL links by creating a new column “eol_uri” based on “eol_id”, using the following expression:

  "http://eol.org/pages/"+value

**Challenges**

Finding the controlled vocabulary to the reconciliation service was difficult, as we are dealing with a very specific area of knowledge I was not familiar with. My first try was the FishBase, which had an API and some services for R set up:

  https://cran.rstudio.com/web/packages/rfishbase/
  https://github.com/ropensci/rfishbase

However, Encyclopedia of Life had a better reconciliation service set up specifically for Open Refine, and URIs with a good landing page with photos. Although it is an encyclopedia and not a controlled vocabulary, it is collaboratively curated and presents up to date information aggregated from several other databases.

As we had many unique species values for this dataset, it was impossible to review all the reconciliation results. I reviewed a sample and, as the results were consistently good, accepted best match suggestions for all remaining entries. It is important to note that sometimes a species may be known by more than one name, so having matches among different names didn’t mean necessarily that the match was wrong (it was usually correct for this dataset).

**Building interface**

- Prepare the data to interface. In order to have links on CARTO interface, we have to add html tags in the source dataset.

  ○ Remove double quotes and language. Create a new column “title” based on the column “http://purl.org/dc/terms/title”, using the following expression:

    value.replace(""", " ").replace("en"," ")

  ○ Add html tags for title links. Create a new column “title_link” based on the column “subject”, using the following expression:
Add html tags for EOL species links. Create a new column “eol_html” based on the column “eol_uri”, using the following expression:

"<a href="+value+">">"+if(isBlank(cells["title"].value), " ", cells["title"].value)+"</a>"

Export the dataset from OpenRefine in .csv format. Name the file “fisheries_cleaned”.

Sign up or Log in to CARTO Builder: [https://carto.com/signup/](https://carto.com/signup/). Create a new map and import the Open Refine exported file to your map.

Georeference your dataset following the instructions here: [https://carto.com/learn/guides/analysis/georeference](https://carto.com/learn/guides/analysis/georeference). Once in your map, click on the dataset on the left menu, then click on “Analysys” > “Add analysys” > Georeference. Select the corresponding column names in your dataset for latitude and longitude. Note that the application is plotting just one resource per location, so we will need to aggregate the results to have all the resources plotted.

Export the georeferenced dataset from CARTO in csv format, in order to incorporate the “the_geom_webmercator” column (with georeferenced data) in your dataset. Name the file “fisheries_cleaned_geo”. Import the dataset back into CARTO map, and delete the previous dataset from your map. This step is necessary since CARTO does not allow georeference analysis and SQL manipulation (that we will need for aggregation) of the data concomitantly.

Click on the dataset on the lateral menu and select the “Data” tab. At the bottom of the lateral panel, enable SQL option. Paste the following query in the editor and click “Apply”:

```
SELECT string_agg(CONCAT(species, ' <br>', eol_html, '<br>'), ' <br>') as new_column_aggregated, title_link, the_geom_webmercator, Min(cartodb_id) cartodb_id
FROM fisheries_cleaned_geo
GROUP BY title_link, the_geom_webmercator
```

Click on the “Pop-up” tab, and enter the following settings:

- **Style**: color
● Window size: 400
○ Header color: #a6e79a
○ Show item: check all boxes (make sure “title_link” is first on the list).

• Click on any point on the map, you should see something similar to this:

![Map Image]

• To build the filters:

  ○ Click on “fisheries_cleaned_geo” dataset, “style” tab, and use the following settings:
    - Aggregation: none
    - Size 6, color #f6ff00
    - Stroke 1 color #FFFFFF, transparent
    - Blending: overlay

  ○ Exit “fisheries_cleaned_geo” dataset and click on the “add” button to re import the same dataset. You will get copy named “fisheries_cleaned_geo_1”. Click on this new dataset, “style” tab, and use the following settings:
    - Aggregation: none
    - Size 12, color #ff0000
    - Stroke 1 color #FFFFFF, A:0.7
    - Blending: none

  ○ Exit the dataset and make it the second one in the list showing on the lateral panel, dragging and dropping it. We want this new layer behind the one that has the pop-up with photos.

  ○ Click on “Widget” tab to add filters. Add widgets following the instructions here: https://carto.com/learn/guides/widgets/exploring-widgets.

• Exit the datasets and change the basemap to “Here” > “Satellite Day”
• Publish your map using the “Publish” button on the left lateral panel, just under the map name. Final result:

**Challenges**

*Finding the right colours for the points on the map was challenging, because of the satellite base map used. Bright colours rendered the best user experience, because of the better contrast against the background.*
References


