Cleaning and Exploring Biomedical Data

With OpenRefine

Dr Jeff Christiansen | 10 July 2014 | v1.2

Table of Contents

1 Overview 2

2 Resources 3

3 Installing Open Refine 4

4 Starting a Project 5

5 Getting Organised 7

6 Exploring the data 9

7 Undo/Redo History 10

8 Filtering the list 11

9 Calling into an API and parsing results 12

10 Calling into another API 16

11 Splitting a column 20

12 Exporting the dataset 21

1. Overview

Open Refine is a powerful free tool for exploring, normalising, cleaning-up and supplementing datasets.

In this tutorial we’ll work through various features of Refine including importing data, faceting, clustering and calling into remote APIs by working on a fictional biomedical research project. We’ll start with a research question in mind and use features of Refine to gain insight and find answers.

The research question relates to genes that are expressed in the 1st branchial arch and optic cup of the mouse embryo at 10.5dpc, and what diseases in humans are associated with the human orthologue of the mouse genes. We’ll start with a list of mouse genes expressed in the 1st branchial arch that we have previously extracted from a public resource (and introduced some inconsistencies into), correct the inconsistencies, filter the list to show only those that are also expressed in the optic cup, and then we’ll supplement the information in the table with both human gene synonyms, and human genetic disorders where there is evidence that there is an underlying molecular basis related to each gene.

1. Resources

Open Refine:

* <http://openrefine.org/>

Open Refine Documentation:

* http://openrefine.org/OpenRefine/documentation
* https://github.com/OpenRefine/OpenRefine/wiki/Documentation-For-Users

The EMAGE mouse gene expression database (source of the original dataset):

* http://www.emouseatlas.org/emage

HGNC database of human gene names:

* <http://www.genenames.org>

Online Mendelian Inheritance in Man (OMIM):

* <http://www.omim.org>

1. Installing Open Refine

To install Open Refine:

1. Go to the main Open Refine website:

* http://openrefine.org/

1. Browse to the Download OpenRefine section.
2. Choose the appropriate download for your operating system. Windows, Mac and Linux are all supported.
3. Follow the installation procedures for your operating system.

.

1. Starting a Project
2. So you have some data to play with in these exercises, we’ve previously searched the EMAGE spatial gene expression database for genes expressed in the region shown in magenta below (i.e. a region corresponding to the 1st branchial arch of the 10.5d*pc* mouse embryo):

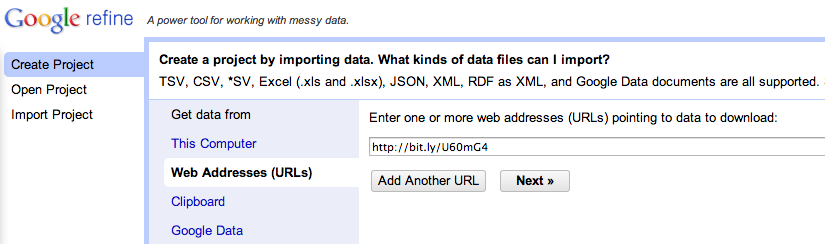


We’ve exported the results in csv (i.e. tabular) format, introduced some anomalies into the data (so you can clean them up in this exercise), and saved them for you here: <http://bit.ly/U60mG4>.

1. Launch Open Refine. It will open in your default web browser. Note: Open Refine is not a cloud application; it runs locally, using your web browser as its primary interface. We recommend using Firefox, Chrome or Safari. To launch OpenRefine in a browser that is not your default, you can copy the URL that appears upon launching in the default browser (e.g. 127.0.0.1:3333 in the following example), and paste this into another browser:



1. Select the **Create Project** tab.
2. Select the **Web Addresses (URLs)** option.
3. Enter http://bit.ly/U60mG4 [[1]](#footnote-1) in the Data file URL field.
4. Click **Next >>**.



1. On the resultant screen, set Project name to 10.5dpc mouse embryo expression:



1. Untick **Parse next \_1\_\_\_ line(s) as column headers**:



1. Click Create Project >>.
2. The project will open with 1036 rows (of which 10 are visible, and you can change to display up to 50 rows):



1. Getting Organised

Renaming columns

1. Select Column 1 menu > Edit column > Rename this column.



1. Rename to *Mouse* Gene Symbol. Click OK
2. Rename Column2 to Embryo age as above.
3. Rename Column3 to Expression text annotation as above.
4. Rename Column4 to EMAGE ID as above.
5. Rename Column5 to Genotype as above.
6. Rename Column6 to Assay type as above.
7. Rename Column7 to Probe ID as above.
8. Rename Column8 to Strain as above.
9. Remove *Column 9* and *Column 10*.

Splitting columns

1. Select Genotype menu > Edit column > Add column based on this column...
2. Set New column name to wt/het.
3. Set Expression to:

if (value != "wild-type", "heterozygote", "wt")

|  |
| --- |
| Open Refine uses GREL (Google Refine Expression Language). More info on GREL syntax can found here <http://code.google.com/p/google-refine/wiki/UnderstandingExpressions>.  The GREL expression used in step 3 is being used to add values to a new column, and these values are based on the existing values found in the genotype column. The genotype column has values that are either “wild-type”, or if the embryo was a heterozygote, the specific genotype is shown (e.g. “Dll3<tm1Rbe>/Dll3<+>”)  The first GREL expression is used to add a column that denotes if the embryos were either wild-type (wt) or heterozygote (het):  **GREL Expression:** **if (value** **!=** **"wild-type",** **"heterozygote", "wt")**  **Meaning:** for every row, **if the value of the cell in the genotype column does not equal “wild-type”, then in the new column add a value of “heterozygote”, otherwise in the new column add a value of “wt”.** |

1. Exploring the data

Sorting Columns

1. Select Genotype menu > Sort
2. Select text and a-z and click OK. All the non wild-type will appear first.

Facet and cluster on Genes

1. Select **Mouse** Gene Symbol menu > Facet > Text Facet. Note that a Gene Symbol facet will appear on the left hand side of the screen. This shows a list of unique gene symbols in the data.
2. Click on 940 choices. A text box will appear so we can copy and paste our list of unique gene symbols into, say, a document.
3. Click count to order to list the most frequently occurring gene symbols first.
4. Click Cluster to reveal and fix some gene symbol nomenclature consistency issues within the dataset that we’ve introduced. Select, for instance nearest neighbor as the method. You'll see that Refine finds some near matches. Now try some of the other methods.
5. You can make your data more consistent by typing the correct value into New Cell Value and ensuring the Merge? checkbox is selected. Use the Merge Selected & Re-Cluster function to actually modify the dataset.
6. Undo/Redo History

Open Refine has an infinite undo history. To access:

1. Click on the Undo/Redo tab. You'll see every action you've done since creating the project.
2. You can undo to any step by clicking on the step you want to revert back to. Similarly, you can redo every step.
3. Click on **Extract** button. The operations you’ve performed can be saved in JSON format so that you have a record of the tasks undertaken and can apply the workflow to other similar data tables in the future.
4. Filtering the list

Because this example list is long, we can filter it and just work on a subset of the data in the list for the next set of operations. *Please ensure you perform this step to save time during the class.*

1. Select **Expression text annotation menu >** **Text filter**
2. Note that an **Expression text annotation** facet will appear on the left hand side of the screen.
3. Type in **optic** and press return
4. Note that now 17 matching rows of the original 1036 are shown that contain the word “optic” (and are selected for the subsequent steps).
5. Calling into an API and parsing results

It's relatively straightforward to draw in data from an external source with Open Refine. In this case we'll call in to HGNC's Human Gene Nomenclature API to get a list of all the human synonyms for the mouse genes in our list.

1. Select **Mouse** Gene symbol menu > Edit column > Add column by fetching URLs.
2. Type HGNC gene info into New column name.
3. Type 20 in Throttle delay.
4. Type the following exactly into the Expression box (including the " symbols):

"http://rest.genenames.org/fetch/symbol/" +value"

|  |
| --- |
| The HGNC API documentation is found here: <http://www.genenames.org/help/rest-web-service-help> where you can see that the **fetch** function is the main request method to retrieve particular records from the HGNC server that will return back all the stored fields. The **fetch** method requires the user to add the queriable field and the query term to the url (e.g. <http://rest.genenames.org/fetch/symbol/ZNF3> - in this case the queriable field is ‘symbol’ and the query term is ‘ZNF3’).  **We can use GREL to construct these URLS for every row in our OpenRefine dataset simply by using the expression:**  "**http://rest.genenames.org/fetch/symbol/**" **+value**"  **This is telling your computer to visit http://rest.genenames.org/fetch/symbol/ and to append the value shown in the gene symbol column for that row to the end of the url.** |

1. You’ll get live feedback in the preview tab so you’ll be able to see if the syntax of the URLs looks correct:



1. Click OK. Open Refine will query the HGNC API for each row in the dataset.
2. In the new column, you should see the results in each row looking something like this:



This is just html/xml but in OpenRefine due to space constraints, it isn’t formatted clearly. To see an example formatted in a structured manner, use Firefox, and paste in <http://rest.genenames.org/fetch/symbol/otx1> and press return. A page like this will be seen showing the hierarchically structured file:



We can now use GREL to parse these results to extract whatever we want from the contents of this file. We’ll extract both the official human gene symbol (approved by the Human Gene Nomenclature Committee), and the VEGA ID (VEGA is a repository for high-quality gene models produced by the manual annotation of vertebrate genomes), and add these to our table.

1. Select HGNC gene info menu > Edit column > Add column based on this column.
2. Type Human Gene Symbol into New column name.
3. Type parseHtml(value).select("str[name=symbol]")[0].htmlText() into **Value**

|  |
| --- |
| **parseHtml(value).select("str[name=symbol]")[0].htmlText()**  **This GREL command parses the html and just selects the value of the string (str) with the name vega\_id, returns the 1st result (if there is more than one) as html text.** |

1. A new column will be added showing the official human gene symbols (e.g. OTX1). Check that there is a one to one correspondence between the mouse and human gene symbols (mouse symbols are denoted in Sentence Case whereas human genes are denoted in UPPERCASE), as the Human gene symbol we have extracted will be used to search the human OMIM database at a later step.

We’ll now also add the VEGA ID for the human gene

1. Select HGNC gene info menu > Edit column > Add column based on this column.
2. Type VEGA ID into New column name.
3. Type parseHtml(value).select("str[name=vega\_id]")[0].htmlText() into **Value**

|  |
| --- |
| **parseHtml(value).select("str[name=vega\_id]")[0].htmlText()**  **This GREL command parses the html and just selects the value of the string (str) with the name vega\_id, returns the 1st result (if there is more than one) as html text.** |

1. A new column will be added showing the VEGA IDs (e.g. OTTHUMG00000129454) if one is found.
2. Select HGNC gene info menu > Edit column > Remove this column.
3. Calling into another API

This time we'll call into OMIM's API to get a list of human genetic disorders that are associated with our list of genes

1. Select **Human** Gene symbol menu > Edit column > Add column by fetching URLs.
2. Type OMIM info into New column name.
3. Type 300 in Throttle delay.
4. Type the following exactly into the Expression box:

"http://www.omim.org/search?index=clinicalSynopsis&  
filter=cs\_molecular\_basis\_exists%3Atrue&search="+value"

|  |
| --- |
| The OMIM API documentation is found here: <http://www.omim.org/help/api>. Note that there are limits in its use: “Entries and clinical synopses are limited to 20 per request if any 'includes' are specified, otherwise there is no limit. The rate of requests is currently limited to 4 requests per second. Your client will be throttled if you exceed the maximum rate, a 409 http status will be returned. Note that repeated requests which get a 409 will lengthen the interval before your access is restored, and if more than thirty 409's are generated your client will be banned for six hours after which access will be restored.”  In this example we want to use the OMIM API to list the genetic disorders that are associated with each gene in our list. This could be done manually on a one-by-one fashion, by visiting the OMIM advanced ‘clinical synopsis’ search page (<http://www.omim.org/search/advanced/clinicalSynopsis>), manually checking the ‘molecular basis’ checkbox and entering a gene symbol as a search term (e.g. Sox9) and pressing search.  If you try this, you’ll see the URL is structured as: <http://www.omim.org/search?index=clinicalSynopsis&start=1&limit=10&search=sox9&sort=score+desc&limit=10&exists=cs_molecular_basis_exists%3Atrue>. In this URL, everything between the ampersand (&) symbols are parameters and their order can be moved around. The above query can also be structured as <http://www.omim.org/search?index=clinicalSynopsis&filter=cs_molecular_basis_exists%3Atrue&search=sox9> (and you can copy and paste this into a browser to see that the same result is returned in both cases).  **We can use GREL to construct URLS in this format for every row in our OpenRefine dataset simply by using the expression:**  **"http://www.omim.org/search?index=clinicalSynopsis&filter=cs\_molecular\_basis\_exists%3Atrue&search="+value"**  **This is telling your computer to visit http://www.omim.org/ and do a search of the clinical synopses data index, and to filter the results to include only those where there is evidence (i.e.true) that a molecular basis exists, and to search for the gene symbol “value”. %3A is the hex code for a colon.** |

1. You’ll get live feedback in the preview so you’ll be able to see if the syntax of the URLs looks correct:  
     
   
2. Click OK. Open Refine will query the API for each row in the dataset.
3. In the new column, you should see the results in each row looking something like this:  
     
     
   

As in the previous example, this is just html but in OpenRefine due to space constraints, it isn’t formatted clearly.

To see an example for one row formatted in a structured manner, use Firefox, and paste in: <http://www.omim.org/search?index=clinicalSynopsis&filter=cs_molecular_basis_exists%3Atrue&search=sox9> and press return to load the results page.

Then select **Tools > Web Developer > Inspector**

In inspector mode, the page will be shown with both the formatted web result page (top panel) and the underlying html code (bottom left). If you hover the cursor over an item of text shown on the top panel (i.e. the nicely formatted web page), you will be shown where this element is found in the underlying html code. For example, in the part of page showing the name of the genetic condition “Campomelic Dysplasia”, this info is within the clinical synopsis results table >tbody > tr > td > a



The meanings of html tags can be found on sites such as <http://www.w3schools.com/tags/>. The tag meanings in this example are <tbody> (table body); <tr> (table row); <td> (table cell); <a> (hyperlinked text).

We can now use GREL to parse the results to extract whatever we want from the contents of this table.

1. Select OMIM menu > Edit column > Add column based on this column.
2. Type value.parseHtml().select("#content table tr td a")[0].htmlText() into **Value**

|  |
| --- |
| **value.parseHtml().select("#content table tr td a")[0].htmlText()**  **This GREL command parses the html to select the value of the 1st occurance of a value ‘a’ in the table, as text.**  **You can try changing the number in square brackets to see how this changes the result (e.g. try entering [6] instead of [0]).** |

1. A new column will be added showing Disease ID and the Disease name (e.g. # 612109. OCULOAURICULAR SYNDROME) if one is found.
2. Select OMIM menu > Edit column > Remove this column.
3. Splitting a column

In this final step, we’ll separate the two values in the Disease column (i.e. ID and name) into 2 separate columns. In this case, they are separated by a full-stop.

1. Select Disease menu > Edit column > Split into several columns.
2. Change the Separator from a comma to a full-stop:



1. Press **OK**.
2. The single Disease column will now be split into 2 columns, one containing the OMIM ID and the other the Disease name, and you can also change the column names accordingly:



1. Exporting the dataset
2. Click Export in the top right-hand corner.
3. Select Excel from the drop-down menu. An Excel dump of your data will be downloaded that can be opened using Excel:



1. Remember that you can also extract the operation history to keep track of how you’ve modified the data, by selecting the **Extract…** option under the **Undo/Redo** tab:



Document Information

|  |  |  |  |
| --- | --- | --- | --- |
| Document Identifier | Open Refine Biomedical.docx | | |
| Document Author | Jeff Christiansen | | |
| Version No. | 1.2 | Version Date | 14 July 2014 |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version No. | Revision Date | Summary of Changes | Revised by |
| 1.1 | 10 July 2014 | Added parsing HGNC xml result to retrieve human gene symbol | Jeff Christiansen |
| 1.2 | 14 July | Corrected several minor typos | Jeff Christiansen |

1. Or https://github.com/IntersectAustralia/TrainingMaterials/blob/master/CleaningAndExploringYourDataWithOpenRefine/Biomed/Data.xls?raw=true [↑](#footnote-ref-1)