Intersect Australia Ltd
PO Box H58, Australia Square
Sydney NSW 1215

T +61 2 8079 2590

ABN 67 131 752 657

INTERSECT

Cleaning and Exploring Biomedical Data

With OpenRefine

Dr Jeff Christiansen | 10 July 2014 | v1.2

Table of Contents

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.

2 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

3 Installing Open Refine

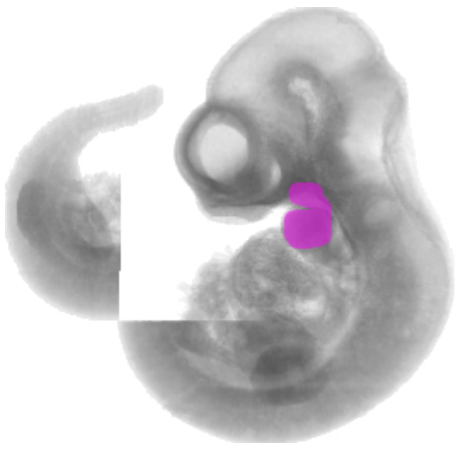
To install Open Refine:

1. Go to the main Open Refine website:
   - http://openrefine.org/
2. Browse to the **Download OpenRefine** section.
3. Choose the appropriate download for your operating system. Windows, Mac and Linux are all supported.
4. Follow the installation procedures for your operating system.

.

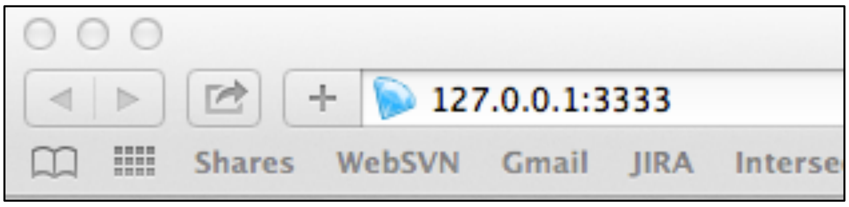
4 Starting a Project

1. 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.

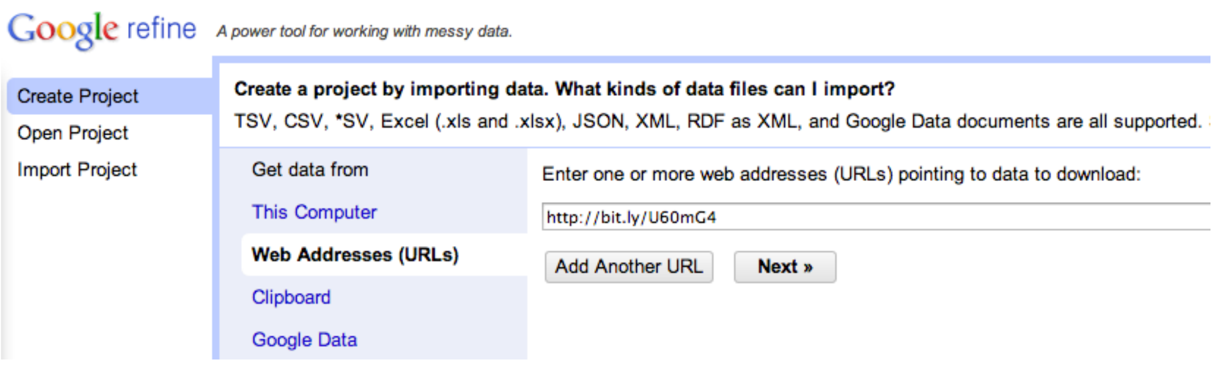
2. Launch Open Refine. It will open in your default web browser. Note: Open Refine is not a cloud application; it runs locally, using your web browser as its primary interface. We recommend using Firefox, Chrome or Safari. To launch OpenRefine in a browser that is not your default, you can copy the URL 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:



3. Select the **Create Project** tab.
4. Select the **Web Addresses (URLs)** option.
5. Enter `http://bit.ly/U60mG4` [1] in the **Data file URL** field.

[1] Or https://github.com/IntersectAustralia/TrainingMaterials/blob/master/CleaningAndExploringYourDataWithOpenRefine/Biomed/Data.xls?raw=true

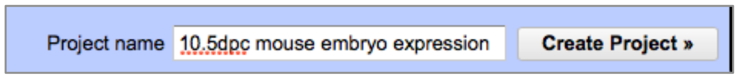
6. Click **Next >>**.



Create a project by importing data. What kinds of data files can I import?
TSV, CSV, *SV, Excel (.xls and .xlsx), JSON, XML, RDF as XML, and Google Data documents are all supported.

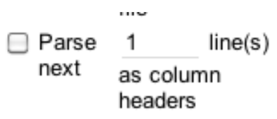
Enter one or more web addresses (URLs) pointing to data to download:
http://bit.ly/U60mG4

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



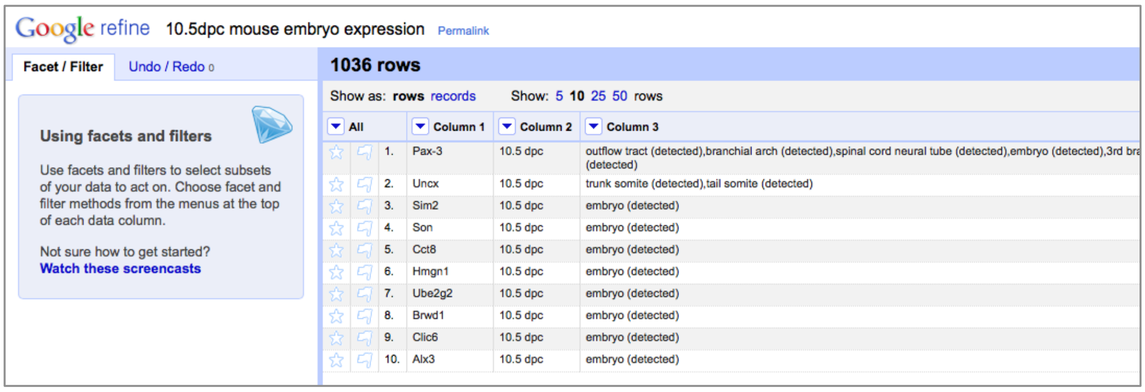
Project name: 10.5dpc mouse embryo expression

8. Untick **Parse next _1___ line(s) as column headers**:



9. Click **Create Project >>**.

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



10.5dpc mouse embryo expression

1036 rows

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

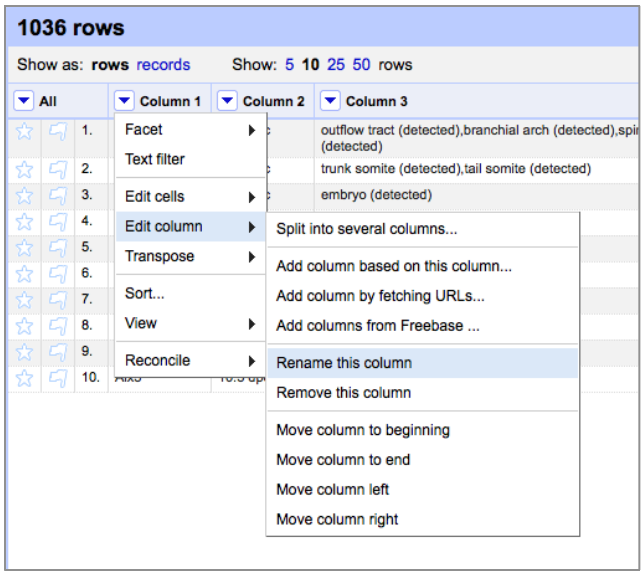
Not sure how to get started?
Watch these screencasts

All	Column 1	Column 2	Column 3
1.	Pax-3	10.5 dpc	outflow tract (detected),branchial arch (detected),spinal cord neural tube (detected),embryo (detected),3rd bra (detected)
2.	Uncx	10.5 dpc	trunk somite (detected),tail somite (detected)
3.	Sim2	10.5 dpc	embryo (detected)
4.	Son	10.5 dpc	embryo (detected)
5.	Cct8	10.5 dpc	embryo (detected)
6.	Hmgn1	10.5 dpc	embryo (detected)
7.	Ube2g2	10.5 dpc	embryo (detected)
8.	Brwd1	10.5 dpc	embryo (detected)
9.	Clic6	10.5 dpc	embryo (detected)
10.	Alx3	10.5 dpc	embryo (detected)

5 Getting Organised

Renaming columns

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



1036 rows

Show as: rows records Show: 5 10 25 50 rows

All	Column 1	Column 2	Column 3
1.			outflow tract (detected),branchial arch (detected),spi (detected)
2.			trunk somite (detected),tail somite (detected)
3.			embryo (detected)

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

6 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.

7 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.

8 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).

9 Calling into an API and parsing results

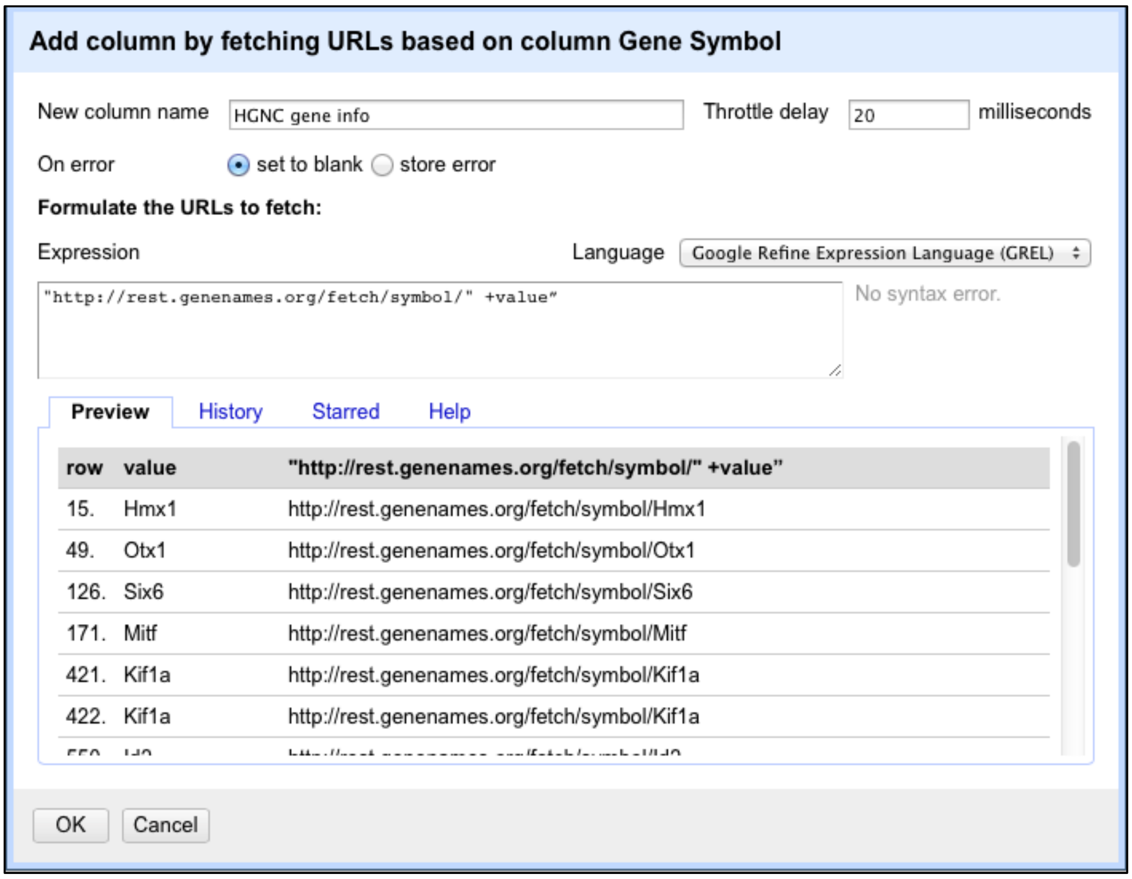
It's relatively straightforward to draw in data from an external source with Open Refine. In this case we'll call in to HGNC's Human Gene Nomenclature API to get a list of all the human synonyms for the mouse genes in our list.

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.**

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



Add column by fetching URLs based on column Gene Symbol

New column name: HGNC gene info
Throttle delay: 20 milliseconds
On error: [✓ set to blank] store error

Formulate the URLs to fetch:

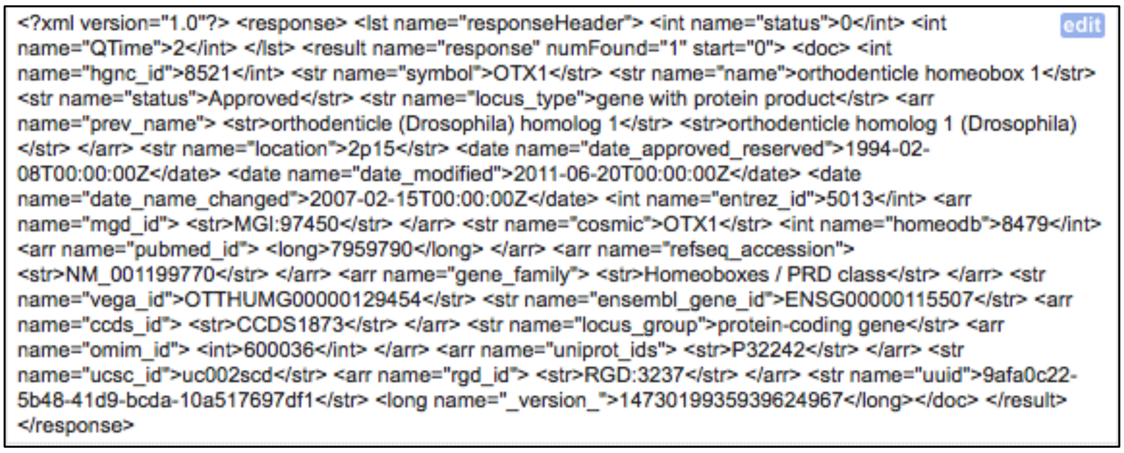
Expression
Language: Google Refine Expression Language (GREL)

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

No syntax error.

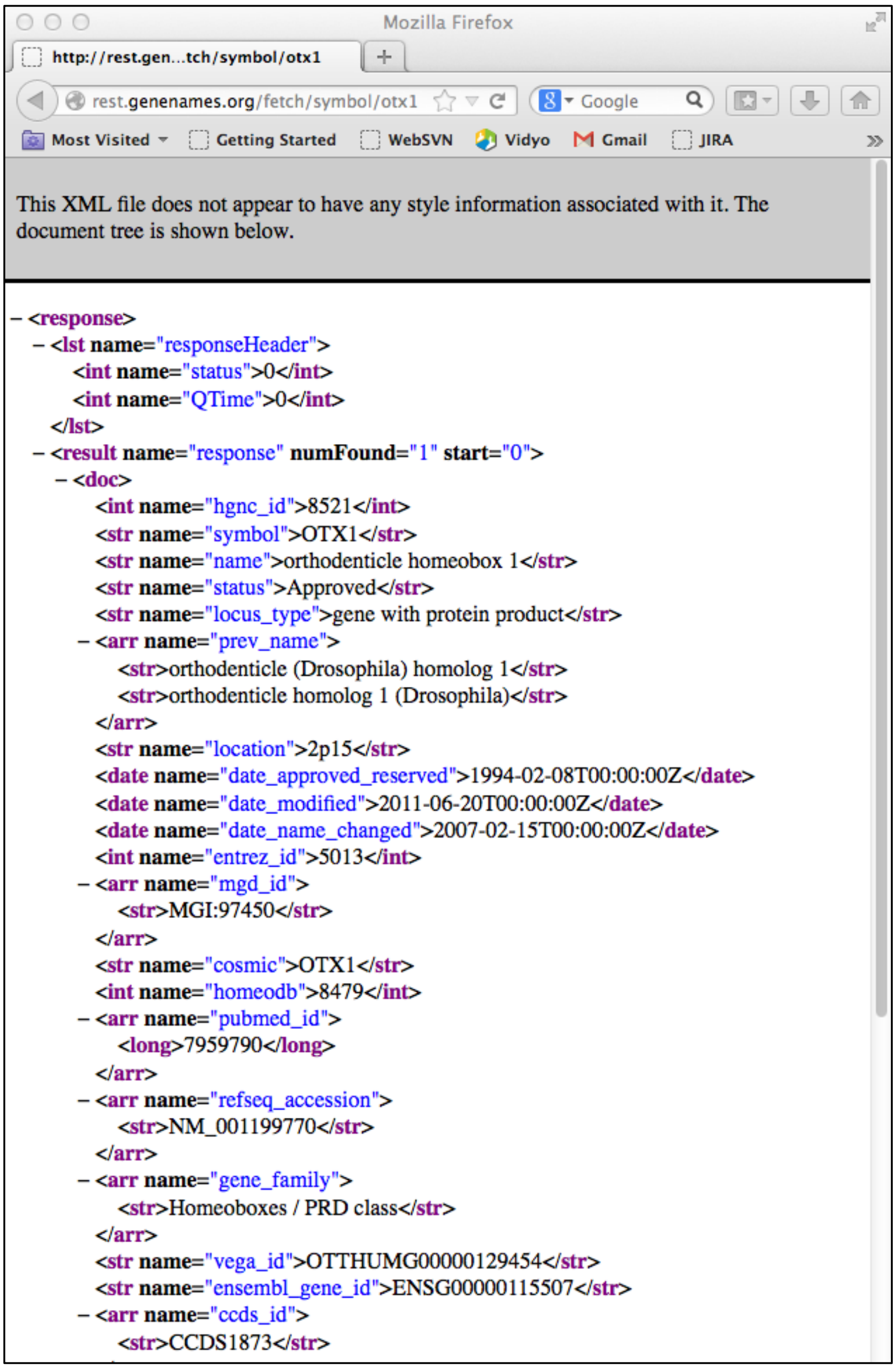
row	value	"http://rest.genenames.org/fetch/symbol/" +value"
15.	Hmx1	http://rest.genenames.org/fetch/symbol/Hmx1
49.	Otx1	http://rest.genenames.org/fetch/symbol/Otx1
126.	Six6	http://rest.genenames.org/fetch/symbol/Six6
171.	Mitf	http://rest.genenames.org/fetch/symbol/Mitf
421.	Kif1a	http://rest.genenames.org/fetch/symbol/Kif1a
422.	Kif1a	http://rest.genenames.org/fetch/symbol/Kif1a

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



<?xml version="1.0"?> <response> <lst name="responseHeader"> <int name="status">0</int> <int name="QTime">2</int> </lst> <result name="response" numFound="1" start="0"> <doc> <int name="hgnc_id">8521</int> <str name="symbol">OTX1</str> <str name="name">orthodenticle homeobox 1</str> <str name="status">Approved</str> <str name="locus_type">gene with protein product</str> <arr name="prev_name"> <str>orthodenticle (Drosophila) homolog 1</str> <str>orthodenticle homolog 1 (Drosophila)</str> </arr> <str name="location">2p15</str> <date name="date_approved_reserved">1994-02-08T00:00:00Z</date> <date name="date_modified">2011-06-20T00:00:00Z</date> <date name="date_name_changed">2007-02-15T00:00:00Z</date> <int name="entrez_id">5013</int> <arr name="mgd_id"> <str>MGI:97450</str> </arr> <str name="cosmic">OTX1</str> <int name="homeodb">8479</int> <arr name="pubmed_id"> <long>7959790</long> </arr> <arr name="refseq_accession"> <str>NM_001199770</str> </arr> <arr name="gene_family"> <str>Homeoboxes / PRD class</str> </arr> <str name="vega_id">OTTHUMG00000129454</str> <str name="ensembl_gene_id">ENSG00000115507</str> <arr name="ccds_id"> <str>CCDS1873</str> </arr> <str name="locus_group">protein-coding gene</str> <arr name="omim_id"> <int>600036</int> </arr> <arr name="uniprot_ids"> <str>P32242</str> </arr> <str name="ucsc_id">uc002scd</str> <arr name="rgd_id"> <str>RGD:3237</str> </arr> <str name="uuid">9afa0c22-5b48-41d9-bcda-10a517697df1</str> <long name="_version_">1473019935939624967</long></doc> </result> </response>

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:


Mozilla Firefox
http://rest.gen...tch/symbol/otx1
rest.genenames.org/fetch/symbol/otx1
Google
Most Visited
Getting Started
WebSVN
Vidyo
Gmail
JIRA

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
– <response>
  – <lst name="responseHeader">
      <int name="status">0</int>
      <int name="QTime">0</int>
    </lst>
  – <result name="response" numFound="1" start="0">
    – <doc>
        <int name="hgnc_id">8521</int>
        <str name="symbol">OTX1</str>
        <str name="name">orthodenticle homeobox 1</str>
        <str name="status">Approved</str>
        <str name="locus_type">gene with protein product</str>
      – <arr name="prev_name">
          <str>orthodenticle (Drosophila) homolog 1</str>
          <str>orthodenticle homolog 1 (Drosophila)</str>
        </arr>
        <str name="location">2p15</str>
        <date name="date_approved_reserved">1994-02-08T00:00:00Z</date>
        <date name="date_modified">2011-06-20T00:00:00Z</date>
        <date name="date_name_changed">2007-02-15T00:00:00Z</date>
        <int name="entrez_id">5013</int>
      – <arr name="mgd_id">
          <str>MGI:97450</str>
        </arr>
        <str name="cosmic">OTX1</str>
        <int name="homeodb">8479</int>
      – <arr name="pubmed_id">
          <long>7959790</long>
        </arr>
      – <arr name="refseq_accession">
          <str>NM_001199770</str>
        </arr>
      – <arr name="gene_family">
          <str>Homeoboxes / PRD class</str>
        </arr>
        <str name="vega_id">OTTHUMG00000129454</str>
        <str name="ensembl_gene_id">ENSG00000115507</str>
      – <arr name="ccds_id">
          <str>CCDS1873</str>
```

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.

8. Select **HGNC gene info menu > Edit column > Add column based on this column.**
9. Type *`Human Gene Symbol`* into **New column name.**
10. 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.
```

11. 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

12. Select **HGNC gene info menu > Edit column > Add column based on this column.**
13. Type *`VEGA ID`* into **New column name.**
14. 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.
```

15. A new column will be added showing the VEGA IDs (e.g. OTTHUMG00000129454) if one is found.
16. Select **HGNC gene info menu > Edit column > Remove this column.**

10 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.

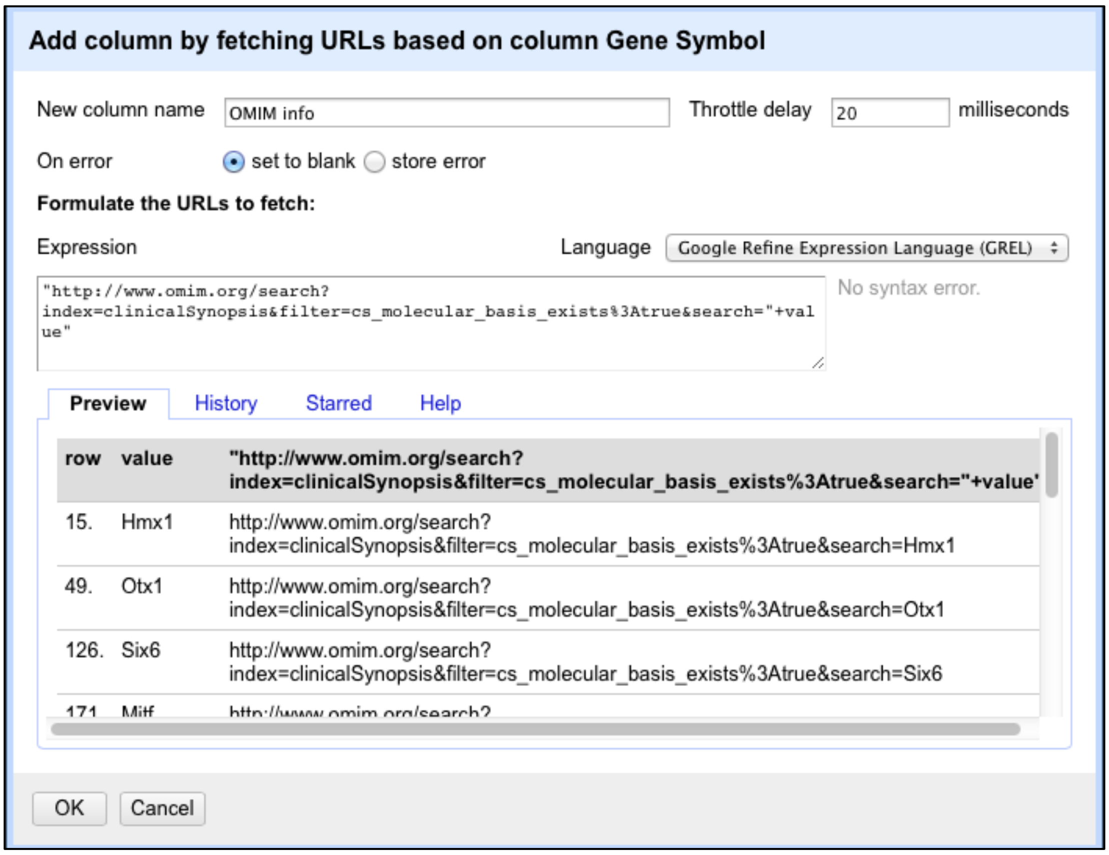
If you try this, you'll see the URL is structured as: http://www.omim.org/search?index=clinicalSynopsis&start=1&limit=10&search=sox9&sort=score+desc&limit=10&exists=cs_molecular_basis_exists%3Atrue. In this URL, everything between the ampersand (&) symbols are parameters and their order can be moved 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.

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



Add column by fetching URLs based on column Gene Symbol

New column name: OMIM info
Throttle delay: 20 milliseconds

On error: [✓ set to blank] store error

Formulate the URLs to fetch:

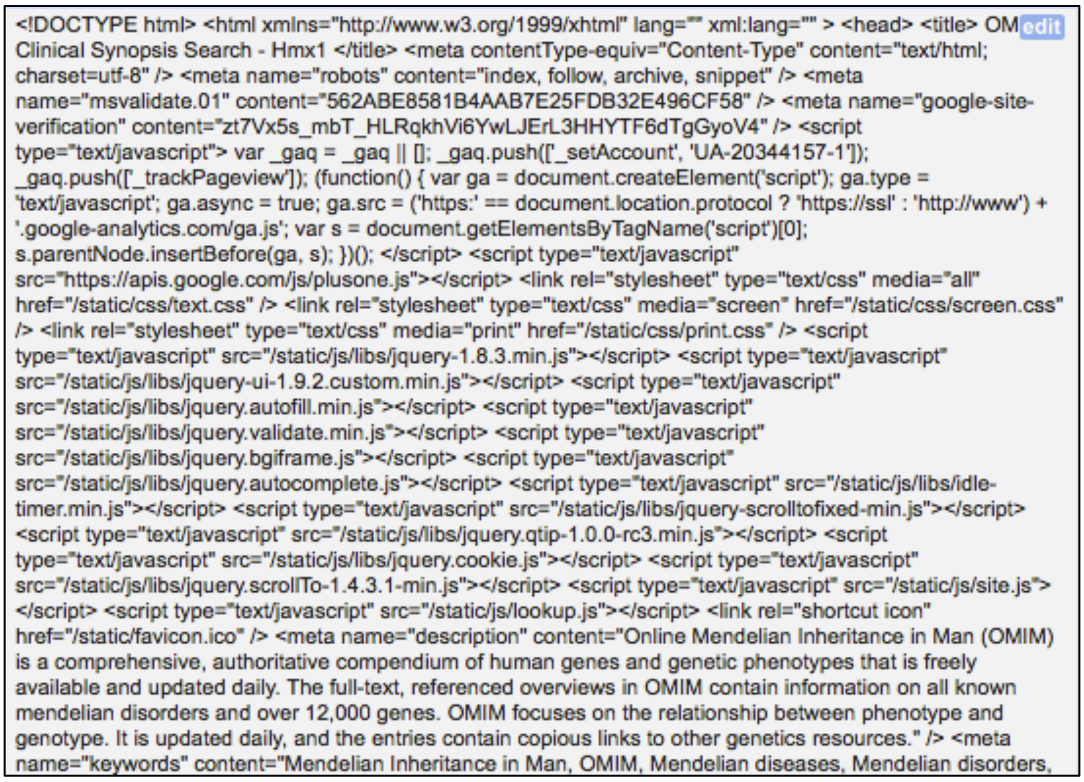
Expression — Language: Google Refine Expression Language (GREL)

```
"http://www.omim.org/search?
index=clinicalSynopsis&filter=cs_molecular_basis_exists%3Atrue&search="+val
ue"
```

No syntax error.

row	value	"http://www.omim.org/search?index=clinicalSynopsis&filter=cs_molecular_basis_exists%3Atrue&search="+value'
15.	Hmx1	http://www.omim.org/search?index=clinicalSynopsis&filter=cs_molecular_basis_exists%3Atrue&search=Hmx1
49.	Otx1	http://www.omim.org/search?index=clinicalSynopsis&filter=cs_molecular_basis_exists%3Atrue&search=Otx1
126.	Six6	http://www.omim.org/search?index=clinicalSynopsis&filter=cs_molecular_basis_exists%3Atrue&search=Six6
171.	Mitf	http://www.omim.org/search?

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



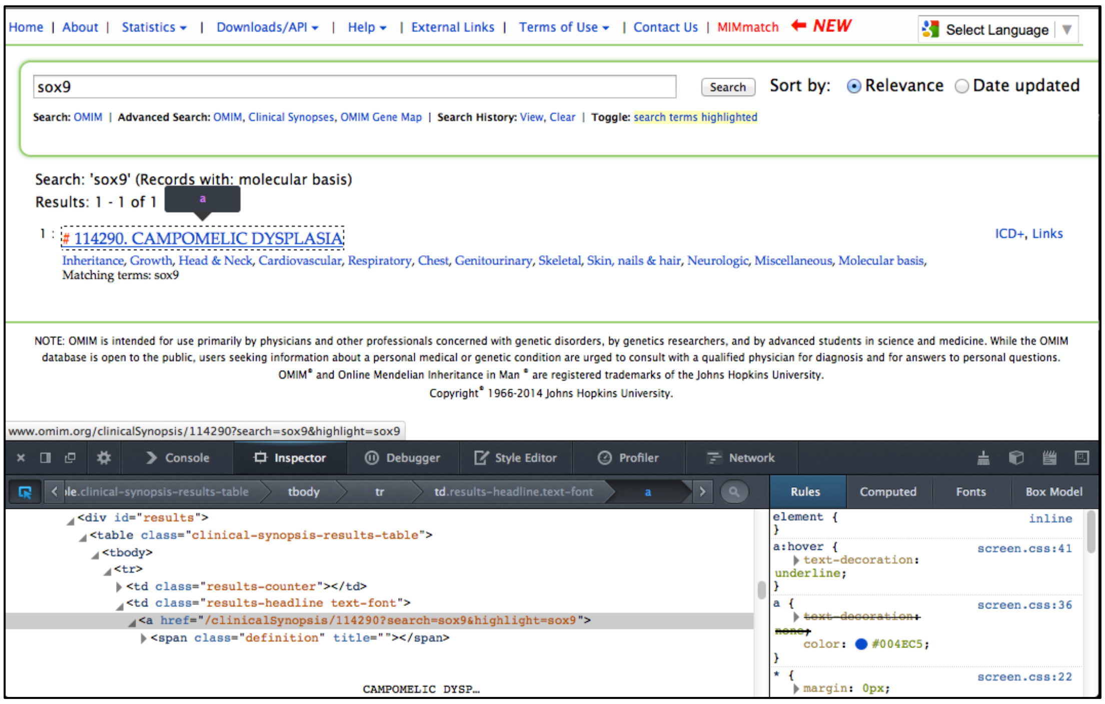
```
<!DOCTYPE html> <html xmlns="http://www.w3.org/1999/xhtml" lang="" xml:lang="" > <head> <title> OMedit
Clinical Synopsis Search - Hmx1 </title> <meta contentType-equiv="Content-Type" content="text/html;
charset=utf-8" /> <meta name="robots" content="index, follow, archive, snippet" /> <meta
name="msvalidate.01" content="562ABE8581B4AAB7E25FDB32E496CF58" /> <meta name="google-site-
verification" content="zt7Vx5s_mbT_HLRqkhVi6YwLJErL3HHYTF6dTgGyoV4" /> <script
type="text/javascript"> var _gaq = _gaq || []; _gaq.push(['_setAccount', 'UA-20344157-1']);
_gaq.push(['_trackPageview']); (function() { var ga = document.createElement('script'); ga.type =
'text/javascript'; ga.async = true; ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') +
'.google-analytics.com/ga.js'; var s = document.getElementsByTagName('script')[0];
s.parentNode.insertBefore(ga, s); })(); </script> <script type="text/javascript"
src="https://apis.google.com/js/plusone.js"></script> <link rel="stylesheet" type="text/css" media="all"
href="/static/css/text.css" /> <link rel="stylesheet" type="text/css" media="screen" href="/static/css/screen.css"
/> <link rel="stylesheet" type="text/css" media="print" href="/static/css/print.css" /> <script
type="text/javascript" src="/static/js/libs/jquery-1.8.3.min.js"></script> <script type="text/javascript"
src="/static/js/libs/jquery-ui-1.9.2.custom.min.js"></script> <script type="text/javascript"
src="/static/js/libs/jquery.autofill.min.js"></script> <script type="text/javascript"
src="/static/js/libs/jquery.validate.min.js"></script> <script type="text/javascript"
src="/static/js/libs/jquery.bgiframe.js"></script> <script type="text/javascript"
src="/static/js/libs/jquery.autocomplete.js"></script> <script type="text/javascript" src="/static/js/libs/idle-
timer.min.js"></script> <script type="text/javascript" src="/static/js/libs/jquery-scrolltofixed-min.js"></script>
<script type="text/javascript" src="/static/js/libs/jquery.qtip-1.0.0-rc3.min.js"></script> <script
type="text/javascript" src="/static/js/libs/jquery.cookie.js"></script> <script type="text/javascript"
src="/static/js/libs/jquery.scrollTo-1.4.3.1-min.js"></script> <script type="text/javascript" src="/static/js/site.js">
</script> <script type="text/javascript" src="/static/js/lookup.js"></script> <link rel="shortcut icon"
href="/static/favicon.ico" /> <meta name="description" content="Online Mendelian Inheritance in Man (OMIM)
is a comprehensive, authoritative compendium of human genes and genetic phenotypes that is freely
available and updated daily. The full-text, referenced overviews in OMIM contain information on all known
mendelian disorders and over 12,000 genes. OMIM focuses on the relationship between phenotype and
genotype. It is updated daily, and the entries contain copious links to other genetics resources." /> <meta
name="keywords" content="Mendelian Inheritance in Man, OMIM, Mendelian diseases, Mendelian disorders,
```

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


Home | About | Statistics | Downloads/API | Help | External Links | Terms of Use | Contact Us | MIMmatch NEW
Select Language
Search
Sort by: [✓ Relevance] Date updated
Search: OMIM | Advanced Search: OMIM, Clinical Synopses, OMIM Gene Map | Search History: View, Clear | Toggle: search terms highlighted
Console
Inspector
Debugger
Style Editor
Profiler
Network
Rules
Computed
Fonts
Box Model

sox9

Search: 'sox9' (Records with: molecular basis)
Results: 1 - 1 of 1

1 : # 114290. CAMPOMELIC DYSPLASIA ICD+, Links
Inheritance, Growth, Head & Neck, Cardiovascular, Respiratory, Chest, Genitourinary, Skeletal, Skin, nails & hair, Neurologic, Miscellaneous, Molecular basis,
Matching terms: sox9

NOTE: OMIM is intended for use primarily by physicians and other professionals concerned with genetic disorders, by genetics researchers, and by advanced students in science and medicine. While the OMIM database is open to the public, users seeking information about a personal medical or genetic condition are urged to consult with a qualified physician for diagnosis and for answers to personal questions.
OMIM® and Online Mendelian Inheritance in Man ® are registered trademarks of the Johns Hopkins University.
Copyright® 1966-2014 Johns Hopkins University.

www.omim.org/clinicalSynopsis/114290?search=sox9&highlight=sox9

le.clinical-synopsis-results-table > tbody > tr > td.results-headline.text-font > a

```
<div id="results">
 <table class="clinical-synopsis-results-table">
  <tbody>
   <tr>
    <td class="results-counter"></td>
    <td class="results-headline text-font">
     <a href="/clinicalSynopsis/114290?search=sox9&highlight=sox9">
      <span class="definition" title=""></span>
CAMPOMELIC DYSP…
```

```
element {                    inline
}
a:hover {                    screen.css:41
  text-decoration:
underline;
}
a {                          screen.css:36
  ~~text-decoration:
none;~~
  color: ● #004EC5;
}
* {                          screen.css:22
  margin: 0px;
```

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.

8. Select **OMIM menu > Edit column > Add column based on this column**.
9. 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.

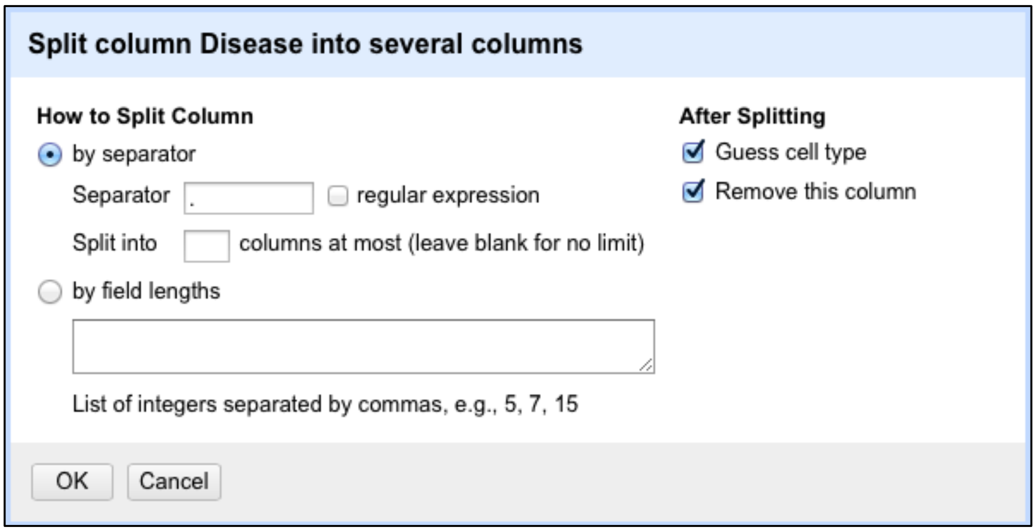
You can try changing the number in square brackets to see how this changes the result (e.g. try entering [6] instead of [0]).

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

11 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:



Split column Disease into several columns

How to Split Column

[✓ by separator]

Separator: . ☐ regular expression

Split into: ___ columns at most (leave blank for no limit)

○ by field lengths

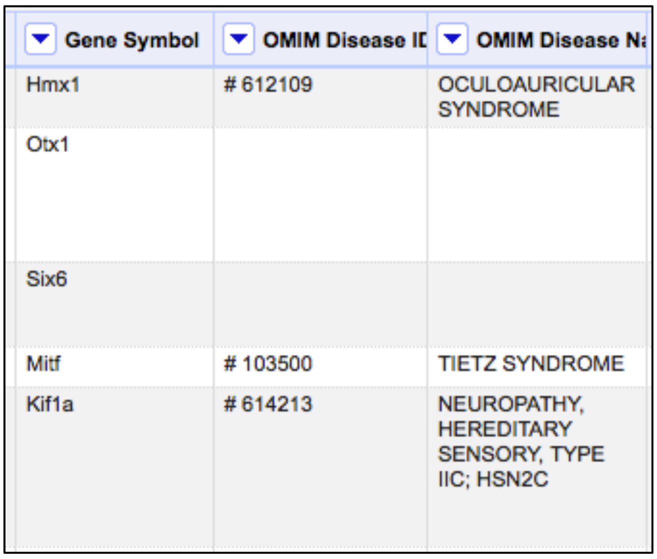
List of integers separated by commas, e.g., 5, 7, 15

After Splitting

☑ Guess cell type

☑ Remove this column

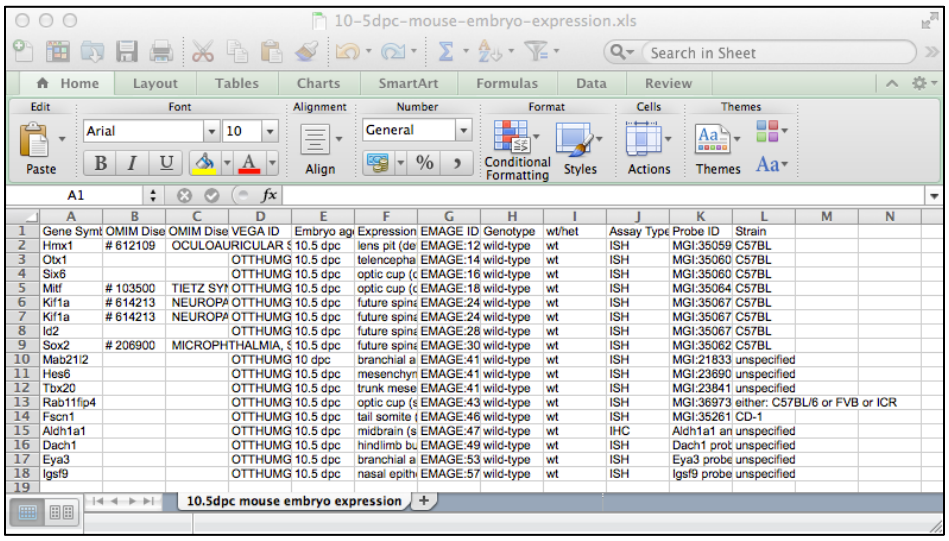
3. Press **OK.**
4. The single Disease column will now be split into 2 columns, one containing the OMIM ID and the other the Disease name, and you can also change the column names accordingly:



Gene Symbol	OMIM Disease ID	OMIM Disease Na
Hmx1	# 612109	OCULOAURICULAR SYNDROME
Otx1		
Six6		
Mitf	# 103500	TIETZ SYNDROME
Kif1a	# 614213	NEUROPATHY, HEREDITARY SENSORY, TYPE IIC; HSN2C

12 Exporting the dataset

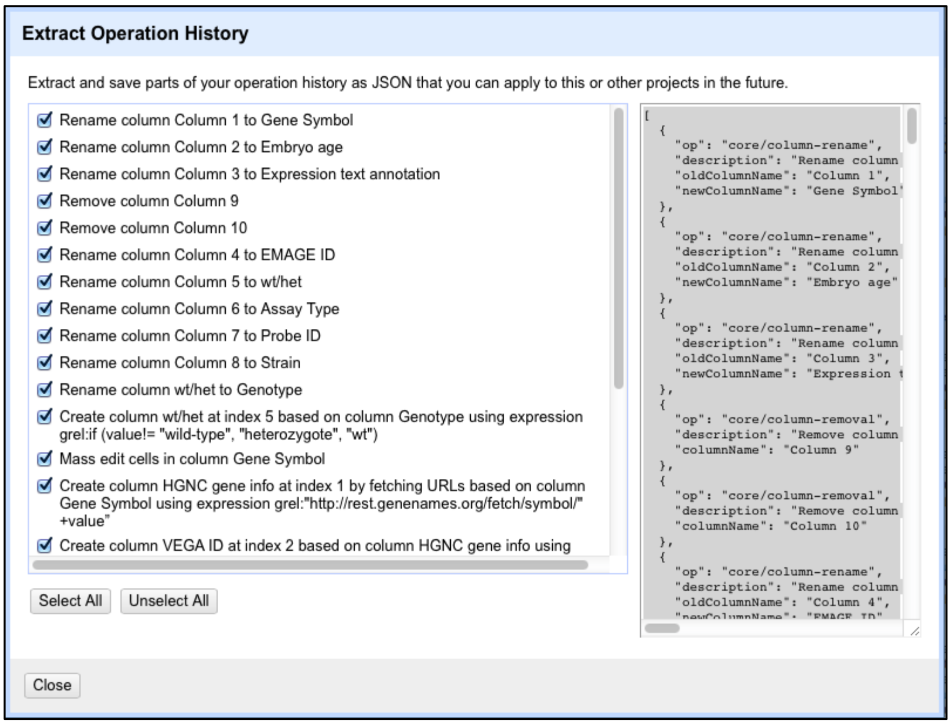
1. Click **Export** in the top right-hand corner.
2. Select **Excel** from the drop-down menu. An Excel dump of your data will be downloaded that can be opened using Excel:


10-5dpc-mouse-embryo-expression.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Gene Symt	OMIM Dise	OMIM Dise	VEGA ID	Embryo agı	Expression	EMAGE ID	Genotype	wt/het	Assay Type	Probe ID	Strain		
2	Hmx1	# 612109	OCULOAURICULAR S		10.5 dpc	lens pit (de	EMAGE:12	wild-type	wt	ISH	MGI:35059	C57BL		
3	Otx1			OTTHUMG	10.5 dpc	telencepha	EMAGE:14	wild-type	wt	ISH	MGI:35060	C57BL		
4	Six6			OTTHUMG	10.5 dpc	optic cup (c	EMAGE:16	wild-type	wt	ISH	MGI:35060	C57BL		
5	Mitf	# 103500	TIETZ SYI	OTTHUMG	10.5 dpc	optic cup (c	EMAGE:18	wild-type	wt	ISH	MGI:35064	C57BL		
6	Kif1a	# 614213	NEUROPA	OTTHUMG	10.5 dpc	future spina	EMAGE:24	wild-type	wt	ISH	MGI:35067	C57BL		
7	Kif1a	# 614213	NEUROPA	OTTHUMG	10.5 dpc	future spina	EMAGE:24	wild-type	wt	ISH	MGI:35067	C57BL		
8	Id2			OTTHUMG	10.5 dpc	future spina	EMAGE:28	wild-type	wt	ISH	MGI:35067	C57BL		
9	Sox2	# 206900	MICROPHTHALMIA, S		10.5 dpc	future spina	EMAGE:30	wild-type	wt	ISH	MGI:35062	C57BL		
10	Mab21l2			OTTHUMG	10 dpc	branchial a	EMAGE:41	wild-type	wt	ISH	MGI:21833	unspecified		
11	Hes6			OTTHUMG	10.5 dpc	mesenchyn	EMAGE:41	wild-type	wt	ISH	MGI:23690	unspecified		
12	Tbx20			OTTHUMG	10.5 dpc	trunk mese	EMAGE:41	wild-type	wt	ISH	MGI:23841	unspecified		
13	Rab11fip4			OTTHUMG	10.5 dpc	optic cup (s	EMAGE:43	wild-type	wt	ISH	MGI:36973	either: C57BL/6 or FVB or ICR		
14	Fscn1			OTTHUMG	10.5 dpc	tail somite (	EMAGE:46	wild-type	wt	ISH	MGI:35261	CD-1		
15	Aldh1a1			OTTHUMG	10.5 dpc	midbrain (s	EMAGE:47	wild-type	wt	IHC	Aldh1a1 an	unspecified		
16	Dach1			OTTHUMG	10.5 dpc	hindlimb bu	EMAGE:49	wild-type	wt	ISH	Dach1 prot	unspecified		
17	Eya3			OTTHUMG	10.5 dpc	branchial a	EMAGE:53	wild-type	wt	ISH	Eya3 probe	unspecified		
18	Igsf9			OTTHUMG	10.5 dpc	nasal epith	EMAGE:57	wild-type	wt	ISH	Igsf9 probe	unspecified		
19														

10.5dpc mouse embryo expression

3. 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:


Extract Operation History

Extract and save parts of your operation history as JSON that you can apply to this or other projects in the future.

- ☑ Rename column Column 1 to Gene Symbol
- ☑ Rename column Column 2 to Embryo age
- ☑ Rename column Column 3 to Expression text annotation
- ☑ Remove column Column 9
- ☑ Remove column Column 10
- ☑ Rename column Column 4 to EMAGE ID
- ☑ Rename column Column 5 to wt/het
- ☑ Rename column Column 6 to Assay Type
- ☑ Rename column Column 7 to Probe ID
- ☑ Rename column Column 8 to Strain
- ☑ Rename column wt/het to Genotype
- ☑ Create column wt/het at index 5 based on column Genotype using expression grel:if (value!= "wild-type", "heterozygote", "wt")
- ☑ Mass edit cells in column Gene Symbol
- ☑ Create column HGNC gene info at index 1 by fetching URLs based on column Gene Symbol using expression grel:"http://rest.genenames.org/fetch/symbol/"+value"
- ☑ Create column VEGA ID at index 2 based on column HGNC gene info using

```
[
  {
    "op": "core/column-rename",
    "description": "Rename column
    "oldColumnName": "Column 1",
    "newColumnName": "Gene Symbol'
  },
  {
    "op": "core/column-rename",
    "description": "Rename column
    "oldColumnName": "Column 2",
    "newColumnName": "Embryo age"
  },
  {
    "op": "core/column-rename",
    "description": "Rename column
    "oldColumnName": "Column 3",
    "newColumnName": "Expression t
  },
  {
    "op": "core/column-removal",
    "description": "Remove column
    "columnName": "Column 9"
  },
  {
    "op": "core/column-removal",
    "description": "Remove column
    "columnName": "Column 10"
  },
  {
    "op": "core/column-rename",
    "description": "Rename column
    "oldColumnName": "Column 4",
    "newColumnName": "EMAGE ID"
```

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