

Introduction to OpenRefine

This Introduction to OpenRefine was developed by Owen Stephens (owen@ostephens.com) on behalf of the British Library in July 2014.

This work is licensed under a Creative Commons Attribution 4.0 International License
<http://creativecommons.org/licenses/by/4.0/>.

It is suggested when crediting this work, you include the phrase “Developed by Owen Stephens on behalf of the British Library”

1. Getting Started

What is OpenRefine?

OpenRefine is described as a tool for working with ‘messy’ data - but what does this mean? It is probably easiest to describe the kinds of data OpenRefine is good at working with and the sorts of problems it can help you solve.

OpenRefine is most useful where you have data in a simple tabular format but with internal inconsistencies either in data formats, or where data appears, or in terminology used. It can help you:

- Get an overview of a data set
- Resolve inconsistencies in a data set
- Help you split data up into more granular parts
- Match local data up to other data sets
- Enhance a data set with data from other sources

Some common scenarios might be:

1. Where you want to know how many times a particular value appears in a column in your data
2. Where you want to know how values are distributed across your whole data set
3. Where you have a list of dates which are formatted in different ways, and want to change all the dates in the list to a single common date format:

Data you have	Desired data
1st January 2014	2014-01-01
01/01/2014	2014-01-01
2014-01-01	2014-01-01
Jan 1 2014	2014-01-01

4. Where you have a list of names or terms that differ from each other but refer to the same people, places or concepts:

Data you have	Desired data
London	London
London]	London
London,]	London
london	London

5. Where you have several bits of data combined together in a single column, and you want to separate them out into individual bits of data with one column for each bit of the data:

Data you have		Desired data							
		Institution	Library name	Address 1	Address 2	Town/City	Region	Country	Postcode
University of Wales, Llyfrgell Thomas Parry Library, Llanbadarn Fawr, ABERYSTWYTH, Ceredigion, SY23 3AS, United Kingdom	University of Wales	Llyfrgell Thomas Parry Library	Llanbadarn Fawr			Aberystwyth	Ceredigion	United Kingdom	SY23 3AS
University of Aberdeen, Queen Mother Library, Meston Walk, ABERDEEN, AB24 3UE, United Kingdom	University of Aberdeen	Queen Mother Library	Meston Walk			Aberdeen		United Kingdom	AB24 3UE
University of Birmingham, Barnes Library, Medical School, Edgbaston, BIRMINGHAM, West Midlands, B15 2TT, United Kingdom	University of Birmingham	Barnes Library	Medical School	Edgbaston	Birmingham	West Midlands	United Kingdom		B15 2TT
University of Warwick, Library, Gibbett Hill Road, COVENTRY, CV4 7AL, United Kingdom	University of Warwick	Library	Gibbett Hill Road		Coventry		United Kingdom		CV4 7AL

6. Where you want to add to your data from an external data source:

Data you have	Date of Birth from VIAF (Virtual International Authority File)	Date of Death from VIAF (Virtual International Authority File)
Braddon, M. E. (Mary Elizabeth)	1835	1915
Rossetti, William Michael	1829	1919
Prest, Thomas Peckett	1810	1879

Downloading OpenRefine

You can download OpenRefine from <http://openrefine.org/download.html>

While the current (as of 10th July 2014) official ‘stable’ version is called ‘Google Refine 2.5’, generally the “OpenRefine 2.6 - Development version” is recommended over this and although labelled ‘beta’ has been stable for some time.

There are versions for Windows, Mac OS X and Linux.

Installing and Running OpenRefine

When you download OpenRefine for Windows or Linux from the address above, you are downloading a zip file. To install OpenRefine you simply unzip the downloaded file wherever you want to install the program. This can be to a personal directory or to an applications or software directory - OpenRefine should run wherever you put the unzipped folder.

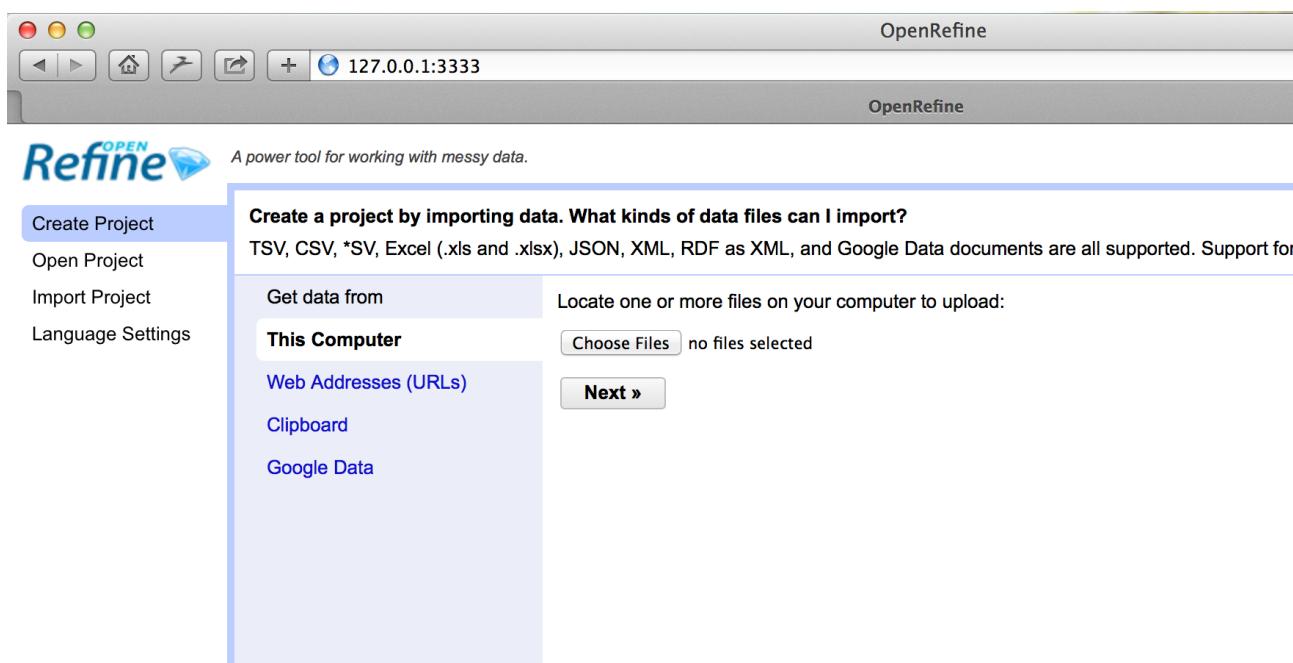
If you are downloading OpenRefine for Mac, you are downloading a ‘dmg’ (disk image) file which you can open, and then drag the OpenRefine application to an appropriate folder on your computer.

OpenRefine is a java application, and you need to have a ‘java runtime environment’ (JRE) installed on your computer to run OpenRefine. If you don’t already have one installed then you can download and install from <http://java.com> by going to the site and clicking ‘Free Java Download’.

To run Refine:

- **On Windows:** Navigate to the folder where you’ve installed OpenRefine and either double-click ‘google-refine.exe’ (for Google Refine 2.5), ‘openrefine.exe’ (for OpenRefine 2.6) or ‘refine.bat’ (for either)
- **On Linux:** Navigate to the folder where you’ve installed OpenRefine in a terminal window and type ‘./refine’
- **On Mac:** Navigate to where you installed OpenRefine and click the OpenRefine icon

The interface to OpenRefine is accessed via a web browser. When you run Refine normally this should open a window in your default web browser pointing at the address <http://127.0.0.1:3333>. If this doesn’t happen automatically you can open a web browser and type in this address.



Getting Help

You can find support, documentation and tutorials on using OpenRefine in various places online including:

- The OpenRefine Wiki <https://github.com/OpenRefine/OpenRefine/wiki>
- The ‘Free your metadata’ site <http://freeyourmetadata.org/>
- The OpenRefine mailing list and forum <http://groups.google.com/d/forum/openrefine>

Exercise 1: Create your first Open Refine project (using provided data)

There are several options for getting your data set into OpenRefine. You can upload or import files in a variety of formats including:

- TSV (tab-separated values)
- CSV (comma-separated values)
- Excel
- JSON (javascript object notation)
- XML
- Google Spreadsheet

To import the data for the exercises below, run OpenRefine and:

- Click ‘Create Project’
- Choose ‘Get Data from this Computer’
- Click ‘Choose Files’
- Locate the file called ‘BL-Flickr-Images-Book-subset.csv’ (this can be downloaded from http://www.meanboyfriend.com/overdue_ideas/wp-content/uploads/2015/02/BL-Flickr-Images-Book-subset.csv)
- Click ‘Next’

You should see a screen as follows:

Identifier	Title	Edition Statement	Place of Publication	Date of Publication
1. 000000206	Walter Forbes. [A novel.] By A. A.		London	1879 [1878]
2. 000000216	All for Greed. [A novel. The dedication signed: A. A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]		London; Virtue & Yorston	1868
3. 000000218	Love the Avenger. By the author of All for Greed. [The dedication signed: A. A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]		London	1869
4. 000000472	Welsh Sketches, chiefly ecclesiastical, to the close of the twelfth century. By the author of Proposals for Christian Union [E. S. A. [i.e. Ernest Appleyard]]		London	1851
5. 000000480	[The World in which I live, and my place in it. By E. S. A. [i.e. Letitia Willgoose Stone.] Edited by ... J. H. Broome.]	A new edition, revised, etc.	London	1857
6. 000000481	[The World in which I live, and my place in it. By E. S. A. [i.e. Letitia Willgoose Stone.] Edited by ... J. H. Broome.]	Fourth edition, revised, etc.	London	1875
7. 000000519	Lagoneils. By the author of Darmayne (F. E. A. [i.e. Florence Emily Ashley])		London	1872
8. 000000667	The Coming of Spring, and other poems. By J. A. [i.e. J. Andrews.]		pp. 40. G. Bryan & Co: Oxford, 1898	

This screen gives you some options to ensure that the data gets imported into OpenRefine correctly. The options vary depending on the type of data you are importing.

In this case you need to:

- Set the ‘Character encoding’ to ‘UTF-8’
- Ensure the first row is used to create the column headings
- Make sure OpenRefine doesn’t try to automatically detect numbers and dates

Once you are happy click ‘Create Project >>’

This will create the project and open it for you. Projects are saved as you work on them, there is no need to save copies as you go along.

To open an existing project in OpenRefine you can click ‘Open Project’ from the main OpenRefine screen (in the lefthand menu). When you click this, you will see a list of the existing projects and can click on a project’s name to open it.

Going Further

Look at the other options on the Import screen - try changing some of these options and see how that changes the Preview and how the data appears after import.

Do you have access to JSON or XML data? If so the first stage of the import process will prompt you to select a ‘record path’ - that is the parts of the file that will form the data rows in the OpenRefine project.

2. Basic OpenRefine Functions

The layout of OpenRefine

OpenRefine displays data in a tabular format. Each row will usually represent a ‘record’ in the data, while each column represents a type of information. This is very similar to how you might view data

The screenshot shows the OpenRefine interface with the following details:

- Header:** Refine OPEN, BL Flickr Images Book subset, Permalink, Open..., Export, Help.
- Facet / Filter:** Facet / Filter, Undo / Redo 0.
- Table Summary:** 8287 rows.
- Show as:** rows records.
- Columns:** Identifier, Title, Edition Statement, Place of Publication, Date of Publication, Publisher, Author, Contributors, Corporate Authors.
- Row Data (Sample):**
 - 1. Identifier: 000000206, Title: Walter Forbes. [A novel.] By A. A., Edition Statement: London, Place of Publication: 1879 [1878], Publisher: S. Tinsley & Co., Author: A. A., Contributors: FORBES, Walter.
 - 2. Identifier: 000000216, Title: All for Greed. [A novel. The dedication signed: A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.], Edition Statement: London; Virtue & Yorston, Place of Publication: 1868, Publisher: Virtue & Co., Author: A. A. A., Contributors: BLAZE DE BURY, Marie Pauline Rose - Baroness
 - 3. Identifier: 000000218, Title: Love the Avenger. By the author of "All for Greed." [The dedication signed: A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.], Edition Statement: London, Place of Publication: 1869, Publisher: Bradbury, Evans & Co., Author: A. A. A., Contributors: BLAZE DE BURY, Marie Pauline Rose - Baroness
 - 4. Identifier: 000000472, Title: Welsh Sketches, chiefly ecclesiastical, to the close of the twelfth century. By the author of "Proposals for Christian Union" (E. S. A. [i.e. Ernest Appleyard]), Edition Statement: London, Place of Publication: 1851, Publisher: James Darling, Author: A. E. S., Contributors: Appleyard, Ernest Silvanus.

in a spreadsheet or database.

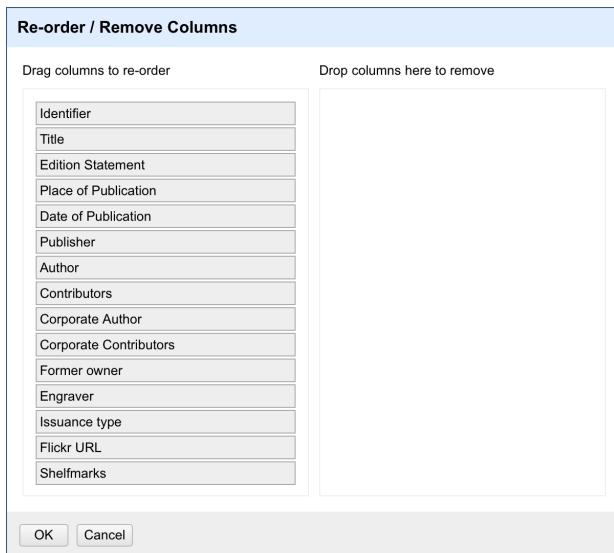
Reordering and renaming columns

Many operations in OpenRefine are accessed from the drop down menus at the top of each column. You can re-order the columns by clicking the drop down menu at the top of the first column (labelled ‘All’), and choosing ‘Edit columns’ -> ‘Re-order / remove columns ...’

The screenshot shows the OpenRefine interface with the following details:

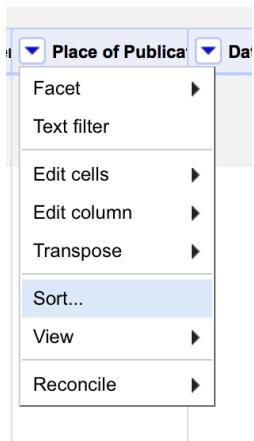
- Header:** Refine OPEN, BL Flickr Images Book subset, Permalink, Open..., Export, Help.
- Facet / Filter:** Facet / Filter, Undo / Redo 0.
- Table Summary:** 8287 rows.
- Show as:** rows records.
- Columns:** Identifier, Title, Edition Statement, Place of Publication.
- Row Data (Sample):**
 - 1. Identifier: 06, Title: Walter Forbes. [A novel.] By A. A., Edition Statement: London.
 - 2. Identifier: 07, Title: All for Greed. [A novel. The dedication signed: A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.], Edition Statement: London; Virtue & Yorston.
 - 3. Identifier: 08, Title: Love the Avenger. By the author of "All for Greed." [The dedication signed: A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.], Edition Statement: London.
- Column Headers (Visible):** Identifier, Title, Edition Statement, Place of Publication.
- Column Headers (Hidden):** Facet, Edit rows, View.
- Open Menu:** The 'Edit columns' menu is open, showing the option 'Re-order / remove columns...'.

You can then drag and drop column names to re-order the columns, or remove columns completely if they are not required:



Sorting data

You can sort data in OpenRefine by clicking on the drop down menu for the column you want to sort on, and choosing 'Sort'



Once you have sorted the data a new ‘Sort’ drop down menu will be displayed:

The screenshot shows the OpenRefine interface with the following details:

- Header:** Refine OPEN BL Flickr Images Book subset Permalink
- Facet / Filter:** Undo / Redo 1
- Row Count:** 8268 rows
- Show as:** rows records (rows is selected)
- Show:** 5 10 25 50 rows
- Sort:** A dropdown menu with a red circle around it.
- Facets and filters section:** Contains a diamond icon and text: "Using facets and filters". It says: "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." It also has a link: "Not sure how to get started? Watch these screencasts".
- Data Preview:** Shows three rows of data:

		Identifier	Title	Edition Statement	Place of Publication
570.	000241463	[The Scornful Ladie ... 1616.]: Single Works	Tenth edition.	London	
6960.	003415351	A poem on the death of Mr. John Philips, etc		Bernard Lintott	

Unlike Excel ‘Sorts’ in OpenRefine are temporary - that is, if you remove the ‘Sort’, the data will go back to its original ‘unordered’ state. The ‘Sort’ drop down menu lets you amend the existing sort (e.g. reverse the sort order), remove existing sorts, and make sorts permanent.

You can sort on multiple columns at the same time.

Exercise 2: Re-order columns and sort data

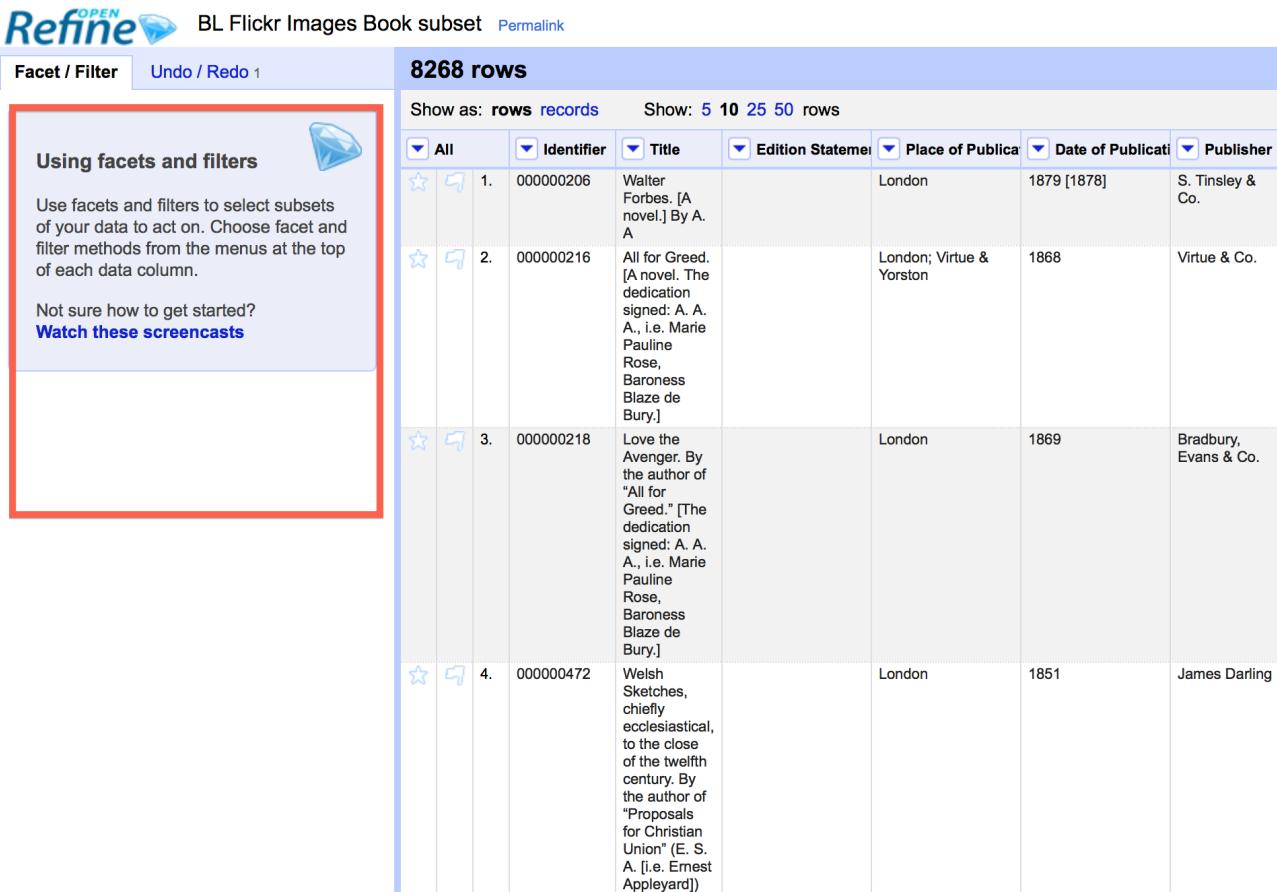
- Find the ‘Date of Publication’ column and sort the information by date of publication
- Move the title column to be the second column in the project (after the “Identifier” column)

Facets

“Facets” are one of the most useful features of OpenRefine and can help both get an overview of the data in a project as well as helping you bring more consistency to the data.

A ‘Facet’ groups all the values that appear in a column, and then allow you to filter the data by these values and edit values across many records at the same time.

The simplest type of Facet is called a ‘Text facet’. This simply groups all the text values in a column and lists each value with the number of records it appears in. The facet information always appears in the left hand panel in the OpenRefine interface.



The screenshot shows the OpenRefine interface with the following details:

- Header:** Refine OPEN, BL Flickr Images Book subset, Permalink
- Left Panel (Facet / Filter):** A red box highlights the "Using facets and filters" section, which contains instructions: "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." It also includes a link: "Not sure how to get started? Watch these screencasts".
- Top Bar:** Facet / Filter, Undo / Redo 1, 8268 rows
- Table Headers:** Show as: rows records, Show: 5 10 25 50 rows, followed by columns: All, Identifier, Title, Edition Statement, Place of Publication, Date of Publication, Publisher.
- Data Rows:** Four rows of data are shown:
 - Row 1: Identifier 000000206, Title "Walter Forbes. [A novel.] By A. A", Place of Publication London, Date of Publication 1879 [1878], Publisher S. Tinsley & Co.
 - Row 2: Identifier 000000216, Title "All for Greed. [A novel. The dedication signed: A. A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]", Place of Publication London; Virtue & Yorston, Date of Publication 1868, Publisher Virtue & Co.
 - Row 3: Identifier 000000218, Title "Love the Avenger. By the author of "All for Greed." [The dedication signed: A. A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]", Place of Publication London, Date of Publication 1869, Publisher Bradbury, Evans & Co.
 - Row 4: Identifier 000000472, Title "Welsh Sketches, chiefly ecclesiastical, to the close of the twelfth century. By the author of "Proposals for Christian Union" (E. S. A. [i.e. Ernest Appleyard])", Place of Publication London, Date of Publication 1851, Publisher James Darling

To create a Text Facet for a column, click on the drop down menu at the top of the column and choose Facet -> Text Facet. The facet will then appear in the left hand panel.

The screenshot below shows a Text Facet on the ‘Issuance Type’ column. You can see this contains two values ‘continuing’ and ‘monographic’. You can filter the data displayed by clicking on one of these headings.

Refine OPEN BL Flickr Images Book subset [Permalink](#)

Facet / Filter Undo / Redo 0

Refresh Reset All Remove All

8287 rows

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

Author	Contributors	Corporate Authc	Corporate Contr	Former owner	Engraver	Issuance type
FORBES, Walter.						monographic
BLAZE DE BURY, Marie Pauline Rose - Baroness						monographic
BLAZE DE BURY, Marie Pauline Rose - Baroness						monographic
Appleyard, Ernest Silvanus.						monographic

Issuance type change

2 choices Sort by: name count Cluster

continuing 19
monographic 8268
Facet by choice counts

You can include multiple values from the facet in a filter at one time by using the ‘Include’ option which appears when you put your mouse over a value in the Facet.

You can also ‘invert’ the filter to show all records which do not match your selected values. This option appears at the top of the Facet panel when you select a value from the facet to apply as a filter.

Filters

As well as using Facets to filter the data displayed in OpenRefine you can also apply ‘Text Filters’ which looks for a particular piece of text appearing in a column. Text filters are applied by clicking the drop down menu at the top of the column you want to apply the filter to and choosing ‘Text filter’.

As with Facets, the Filter options appear in the left hand panel in OpenRefine. Simply type in the text you want to use in the Filter to display only rows which contain that text in the relevant column.

The screenshot shows the OpenRefine interface with a dataset titled "BL Flickr Images Book subset". On the left, there's a facet/filter sidebar with a "Place of Publication" facet set to "London". The main area displays 4219 matching rows out of 8287 total. The columns are Identifier, Title, Edition Statement, Place of Publication, and Date of Publication. The first row shows "000000206" as the Identifier, "Walter Forbes. [A novel.] By A. A." as the Title, and "London" as the Place of Publication. The Date of Publication is "1879 [1878]". The second row shows "000000216" as the Identifier, "All for Greed. [A novel. The dedication signed: A. A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]" as the Title, and "edit" as the Edition Statement. The Place of Publication is "London; Virtue & Yorston" and the Date of Publication is "1868". The third row shows "000000218" as the Identifier, "Love the Avenger. By the author of "All for Greed." [The dedication signed: A. A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]" as the Title, and "London" as the Place of Publication. The Date of Publication is "1869".

Identifier	Title	Edition Statement	Place of Publication	Date of Publication
000000206	Walter Forbes. [A novel.] By A. A.		London	1879 [1878]
000000216	All for Greed. [A novel. The dedication signed: A. A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]	edit	London; Virtue & Yorston	1868
000000218	Love the Avenger. By the author of "All for Greed." [The dedication signed: A. A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]		London	1869

Working with filtered data

It is very important to note that when you have filtered the data displayed in OpenRefine, any operations you carry out will apply only to the rows that match the filter - that is the data currently being displayed.

In particular if you wish to remove rows that match a filter, you can do this as follows:

- Filter the data using Facets or Filters
- Clicking on the drop down menu under the ‘All’ column heading (this is always the first column in an OpenRefine project)
- Choose ‘Edit rows’ -> ‘Remove all matching rows’

This will remove all rows that were displayed by the filter:

The screenshot shows the Refine interface with the following details:

- Facet / Filter**: Shows a facet for **Issuance type** with **continuing** selected.
- Undo / Redo**: Shows 0 undo/redo steps.
- Rows**: 8287 rows.
- Show as:** rows records
- Show:** 5 10 25 50 rows
- Facet**: Shows a table with columns for Identifier, Title, Edition Statement, and Place of Publication.
- Actions (Context Menu):** Options include **Star rows**, **Unstar rows**, **Flag rows**, **Unflag rows**, and **Remove all matching rows**.

Exercise 3: Remove all ‘continuing’ publications from this data set

- Create a facet for the ‘Issuance Type’ column
- Filter the data to only show those rows which represent ‘continuing’ publications
- Remove these rows from the data set

Going Further

Create a text facet for the Date of Publication column. What are the problems with this facet?

Using the facet, how would you find the most common value in the Date of Publication column?

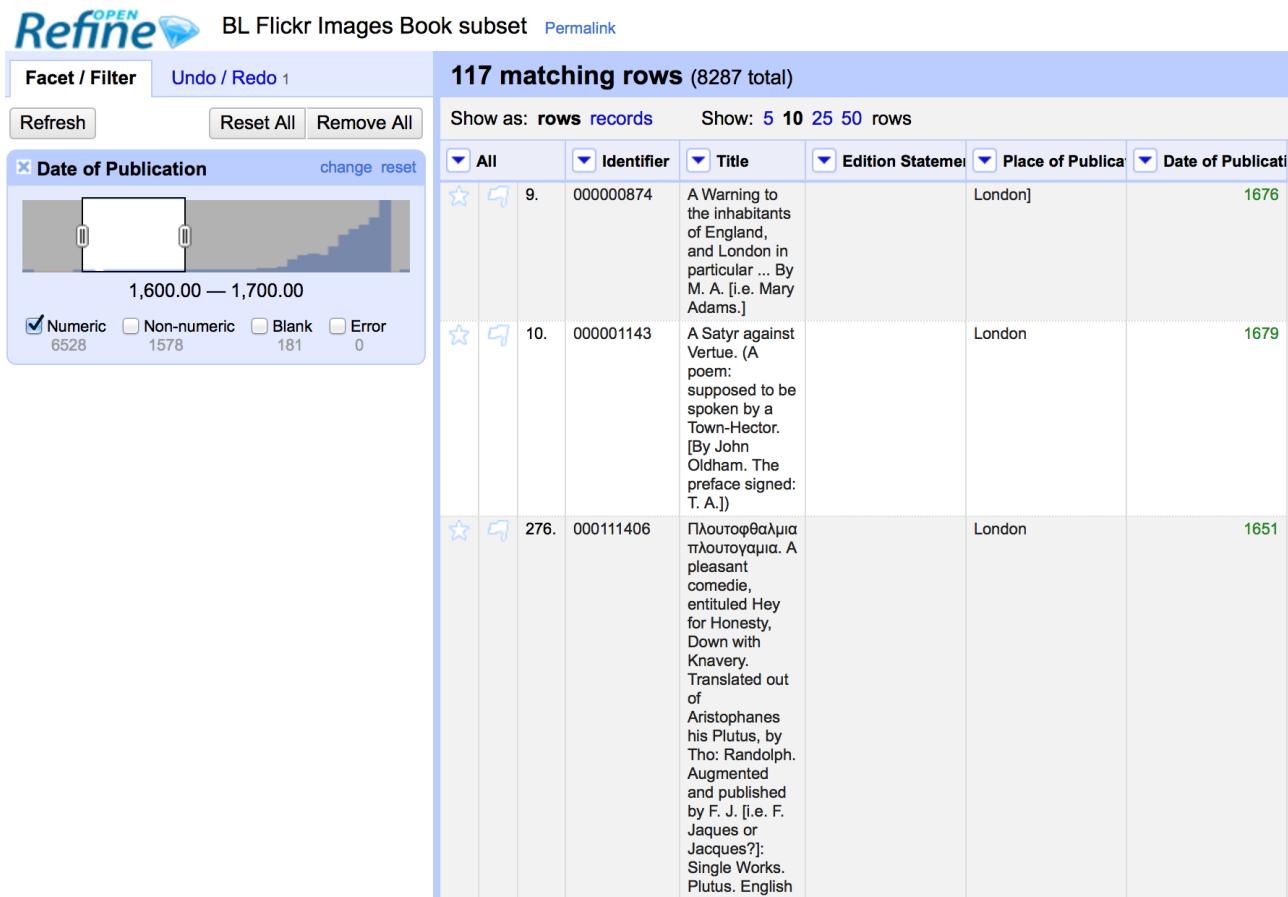
What happens to the “Date of Publication” facet if you apply a text filter to the “Place of Publication column?”

More on Facets

As well as ‘Text facets’ Refine also supports a range of other types of facet. These include:

- Numeric facets
- Timeline facets (for dates)
- Custom facets
- Scatterplot facets

Numeric and Timeline facets display graphs instead of lists of values. The graph includes ‘drag and drop’ controls you can use to set a start and end range to filter the data displayed.



Scatterplot facets are less commonly used - for further information on these see the tutorial at http://enipedia.tudelft.nl/wiki/OpenRefine_Tutorial#Exploring_the_data_with_scatter_plots

Custom facets are a range of different types of facets, and also allow you write your own custom facets. Some of the default custom facets are:

- Word facet - this breaks down text into words and counts the number of records each word appears in
- Duplicates facet - this results in a binary facet of ‘true’ or ‘false’. Rows appear in the ‘true’ facet if the value in the selected column is an exact match for a value in the same column in another row
- Text length facet - creates a numeric facet based on the length (number of characters) of the text in each row for the selected column. This can be useful for spotting incorrect or unusual data in a field where specific lengths are expected (e.g. if the values are expected to be years, any row with a text length more than 4 for that column is likely to be incorrect)

- Facet by blank - a binary facet of ‘true’ or ‘false’. Rows appear in the ‘true’ facet if they have no data present in that column. This is useful when looking for rows missing key data.

Facets are intended to group together common values and OpenRefine limits the number of values allowed in a single facet to ensure the software does not perform slowly or run out of memory. If you create a facet where there are many unique values (for example, a facet on a ‘book title’ column in a data set that has one row per book) the facet created will be very large and may either slow down the application, or OpenRefine will not create the facet.

Exercise 4: Find all publications without a date of publication

- Use the ‘Facet by blank’ function to find all publications in this data set without a date of publication
- Filter the data to only those without a date of publication - do you notice anything about these records? [Hint: look at the “Place of Publication” column]

Amending data through facets

If you create a text facet you can edit the values in the facet to change the value for several records at the same time. To do this, simply mouse-over the value you want to edit and click the ‘edit’ option that appears:

Edition Statement	Count
[Another edition.]	115
Second edition.	31
Third edition.	19
New edition.	17
A new edition.	9
[Another edition.] With illustrations by H. K. Browne.	6
Fourth edition.	6
New edition, etc.	5
The second edition.	5
Ninth edition.	4
Sixth edition.	4

This approach is useful in relatively small facets where you might have small variations through punctuation or typing errors etc. For example, a column that should contain only terms from a small restricted list such as days of the week or months of the year.

The list of values in the facet will update as you make edits.

Exercise 5: Edit edition statements via a text facet

- Create a text facet for the Edition column
- Sort the facet by ‘count’ to see the most common values
- Pick the facets that refer to a ‘second edition’ and edit them using a consistent wording

Using Clustering to find similar values

Another function that is provided with facets is the ‘Cluster’ function. The Cluster function looks for similar values across the facet and enables you to merge together several facets to a single value.

This is very effective where you have data where there can be minor variations in data values that are likely such as names of people, organisations and places.

To use the 'Cluster' function, create a Facet on the relevant column and click the 'Cluster' button. This will bring up a new window where you can see the 'Clusters' that have been detected and work with them:

Cluster & Edit column "Publisher"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "GÄ¶del" and "Godel" probably refer to the same person. [Find out more ...](#)

Method Keying Function 38 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
5	32	<ul style="list-style-type: none"> Printed for private circulation (25 rows) [Printed for private circulation (3 rows) [Printed for private circulation] (2 rows) [Printed for Private Circulation (1 rows) printed for private circulation (1 rows) 	<input type="checkbox"/>	Printed for private circulation
4	50	<ul style="list-style-type: none"> Smith, Elder & Co. (47 rows) Smith Elder & Co. (1 rows) Smith, Elder Co. (1 rows) Smith, Elder, & Co. (1 rows) 	<input type="checkbox"/>	Smith, Elder & Co.
4	63	<ul style="list-style-type: none"> Privately printed (53 rows) [Privately printed (4 rows) [Privately printed] (4 rows) Privately printed. (2 rows) 	<input type="checkbox"/>	Privately printed
3	88	<ul style="list-style-type: none"> Macmillan & Co. (85 rows) Macmillan Co. (2 rows) MacMillan & Co. (1 rows) 	<input type="checkbox"/>	Macmillan & Co.
2	13	<ul style="list-style-type: none"> Clarendon Press (12 rows) [Clarendon Press (1 rows) 	<input type="checkbox"/>	Clarendon Press

Choices in Cluster

2 — 5

Rows in Cluster

2 — 88

Average Length of Choices

6 — 33

Length Variance of Choices

0 — 6

The 'Clusters' are created automatically according to an algorithm. There are a number of different algorithms supported by OpenRefine - some experimentation maybe required to see which clustering algorithm works best with any particular set of data, and you may find that using different algorithms highlights different clusters.

For more information on the methods used to create Clusters see <https://github.com/OpenRefine/OpenRefine/wiki/Clustering-In-Depth>

For each cluster you have the option of 'merging' the values together - that is replace with a single consistent value. By default OpenRefine uses the most common value in the cluster as the new value, but you can select one of the other values by clicking the value itself, or you can simply type the desired value into the 'New Cell Value' box.

The Clustering function can also be accessed via the drop down menu at the top of a column by selecting "Edit cells" -> "Cluster and edit ..."

Exercise 6: Use Clustering to clean up publisher data

- Create a Text Facet for the "Publisher" column
- Click 'Cluster'
- Using the 'key collision' method with the 'fingerprint' Keying Function work through the clusters of values, merging them to a single value where appropriate

Going Further

Experiment with other clustering methods - which do you think gives the best results? What is the downside of using 'nearest neighbour' methods with data such as that in the 'Publisher' column?

Try using Clustering with other columns such as 'Place of Publication' and 'Date of Publication'

Introducing Transformations

Through facets, filters and clusters OpenRefine offers relatively straightforward ways of getting an overview of your data, and making changes where you want to standardise terms used to a common set of values.

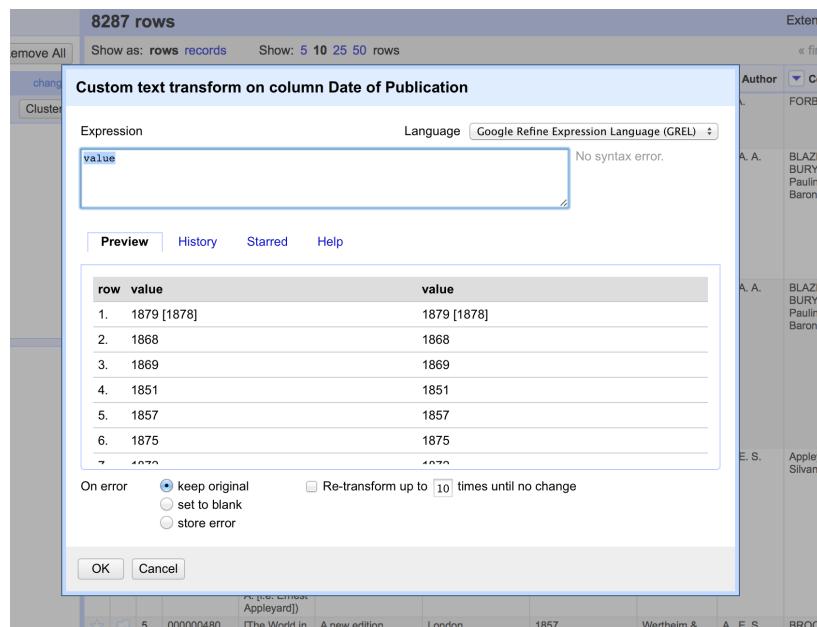
However, sometimes there will be changes you want to make to the data that cannot be achieved in this way. Such types of changes include:

- Splitting data that is in a single column into multiple columns (e.g. splitting an address into multiple parts)
- Standardising the format of data in a column without changing the values (e.g. removing punctuation or standardising a date format)
- Extracting a particular type of data from a longer text string (e.g. finding ISBNs in a bibliographic citation)

To support this type of activity OpenRefine supports ‘Transformations’ which are ways of manipulating data in columns. Transformations are normally written in a special language called ‘GREL’ (Google Refine Expression Language). To some extent GREL expressions are similar to Excel Forumla, although they tend to focus on text manipulations rather than numeric functions.

Full documentation for the GREL is available at <https://github.com/OpenRefine/OpenRefine/wiki/Google-refine-expression-language>. This tutorial covers only a small subset of the commands available.

To start writing transformations, select the column on which you wish to perform a transformation and choose “Edit cells” -> “Transform...” to see the following screen:



In this screen you have a place to write a transformation (the ‘Expression’ box) and then the ability to Preview the effect the transformation would have on the first 10 rows of your data.

The transformation you type into the ‘Expression’ box has to be a valid GREL expression. The simplest expression is simply the word ‘value’ by itself - which simply means ‘the value that is currently in the column’ - that is, “make no change”.

GREL functions are written by giving a value of some kind (a text string, a date, a number etc.) to a GREL function. Some GREL functions take additional parameters or options which control how the function works. GREL supports two syntaxes:

`value.function(options)`
`function(value, options)`

Either is valid, and which is used is completely down to personal preference.

Next to the ‘Preview’ option are options to view:

- History - a list of transformations you’ve previously used with the option to reuse them immediately or to ‘star’ them for easy access
- Starred - a list of transformations you’ve ‘starred’ via the ‘History’ view
- Help - a list of all the GREL functions and brief information on how to use them

Some simple transformations

GREL expression	Examples	Action carried out
<code>toUppercase(string)</code>	<code>toUppercase(value)</code> <code>value.toUppercase()</code>	Converts the current value to uppercase
<code>toLowercase(string)</code>	<code>toLowercase(value)</code> <code>value.toLowerCase()</code>	Converts the current value to lowercase
<code>toTitlecase(string)</code>	<code>toTitlecase(value)</code> <code>value.toTitlecase()</code>	Converts the current value to titlecase (i.e. each word starts with an uppercase character and all other characters are converted to lowercase)
<code>trim(string)</code>	<code>trim(value)</code> <code>value.trim()</code>	Removes any ‘whitespace’ characters (e.g. spaces, tabs) from the start or end of the current value
<code>substring(string, number from, optional number to)</code>	<code>substring(value, 0, 4)</code> <code>value.substring(0,4)</code>	Finds the first four characters of the current value
<code>replace(string, string to find, replacement string)</code>	<code>replace(value, “a”, “b”)</code> <code>value.substring(“a”, “b”)</code>	Find the letter ‘a’ in the current value and replace it with the letter ‘b’
<code>+</code>	“Prefix: ” + value	Adds (concatenates) the word “Prefix” to the front of the current value

Exercise 7: Cleaning up Date of Publication using simple transformations

- Create a facet based on the Date of Publication column
- Sort the facet by ‘name’
- What are common issues with the values in this list?
- Use the ‘replace’ GREL expression to remove the characters [,] and ? from the Date of Publication column

Going Further

What other issues can you see with the date of publication column?

What approaches might you use to overcome these issues?

Do you think it is possible to achieve a single column with a four digit date in it to represent date of publication?

Undo and Redo

OpenRefine lets you undo, and redo, any number of steps you have taken in cleaning the data. This means you can always try out transformations and ‘undo’ if you need to. The way OpenRefine records the steps you have taken even allows you to take the steps you’ve carried out on one data set, and apply it to another data set by a simple copy and paste operation.

The ‘Undo’ and ‘Redo’ options are accessed via the lefthand panel.

The screenshot shows the OpenRefine interface with the following details:

- Header:** Refine OPEN, BL Flickr Images Book subset, Permalink
- Left Panel (Facet / Filter):** Shows a list of steps:
 0. Create project
 1. Mass edit 544 cells in column Publisher
 2. Text transform on 1128 cells in column Date of Publication: grel:value.replace(/\[\!\?\]/,"")
- Central Area:** Title: 8287 rows. Show as: rows records. Show: 5 10 25 50 rows. A table preview is shown with four rows of data:

	All	Identifier	Title	Edition Statement	Place of Publication	Date of Publication
1.	000000206	Walter Forbes. [A novel.] By A. A.			London	1879 1878
2.	000000216	All for Greed. [A novel. The dedication signed: A. A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]			London; Virtue & Yorston	1868
3.	000000218	Love the Avenger. By the author of "All for Greed." [The dedication signed: A. A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]			London	1869
4.	000000472	Welsh Sketches, chiefly ecclesiastical, to the close of the twelfth century. By the author of "Proposals for Christian Union" (E. S. A. [i.e. Ernest Appleyard])			London	1851

The Undo/Redo panel lists all the steps you’ve taken so far. To undo steps, simply click on the last step you want to preserve in the list and this will automatically undo all the changes made since that step.

The remaining steps will continue to show in the list but greyed out, and you can reapply them by simply clicking on the last step you want to apply.

However, if you ‘undo’ a set of steps and then start doing new transformations, the greyed out steps will disappear and you will no longer have the option to ‘redo’ these steps.

If you wish to save a set of steps to be re-applied later, or to a different project, you can click the ‘Extract’ button. This gives you the option to select the steps you want to save, and to copy the transformations included in the selected steps in a format called ‘JSON’

To apply a set of steps you have copied or saved in this ‘JSON’ format use the ‘Apply’ button and paste in the JSON. In this way you can share transformations between projects and each other.

Undo/Redo data is stored with the Project and is saved automatically as you work, so next time you open the project, you can access your full history of steps you have carried out and undo/redo in exactly the same way.

Exporting data

Once you have finished working with a data set in OpenRefine you may wish to export it. The export options are accessed through the ‘Export’ button at the top right of the OpenRefine interface

Identifier	Title	Edition Statement	Place of Publication	Date of Publication	Publisher
000000206	Walter Forbes. [A novel.] By A. A.		London	1879 [1878]	S. Tinsley & Co.
000000216	All for Greed. [A novel. The dedication signed: A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]		London; Virtue & Yorston	1868	Virtue & Co.
000000218	Love the Avenger. By the author of "All for Greed." [The dedication signed: A. A., i.e. Marie Pauline Rose, Baroness Blaze de Bury.]		London	1869	Bradbury, Evans & Co.
000000472	Welsh Sketches, chiefly		London	1851	James Darling
					A. E. S.
					Appleyard, Ernest Silvanus.

Export formats support include HTML, Excel and comma- and tab-separated value (csv and tsv). You can also write a custom export, selecting to export specific fields, adding a header or footer and specifying the exact format.

3. Data types and Regular Expressions

Understanding data types and regular expressions will help you write more complex transformations using GREL.

Data types in OpenRefine

Every piece of data in OpenRefine is has a ‘type’. The most common ‘type’ is a ‘string’ - that is a piece of text. However there are other data types available and transformations let you convert data from one type to another where appropriate. The data types supported are:

- String
- Number
- Date
- Boolean
- Array

The first three are hopefully self-explanatory, but the latter two may require slightly more explanation.

A ‘**Boolean**’ is a binary value that can either be ‘true’ or ‘false’. Boolean values can be used directly in OpenRefine cell, but is more often used in transformations as part of a GREL expression. For example:

```
value.contains("test")
```

Generates a boolean value of either ‘true’ or ‘false’ depending on whether the current value in the cell contains the text ‘test’ anywhere. Such tests can be combined with other GREL expressions to create more complex transformations.

An ‘**Array**’ is a list of values, represented in Refine by the use of square brackets containing a list of values surrounded by inverted commas and separated by commas. For example an array listing the days of the week would look like:

```
["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"]
```

Arrays can be sorted, de-duplicated, and manipulated in other ways in GREL expressions, but cannot appear directly in an OpenRefine cell. Arrays in OpenRefine are usually the result of a transformation. For example the ‘split’ function takes a string, and changes it into an array based on a ‘separator’. For example if a cell has the value:

“Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday”

This can be transformed into an array using the ‘split’ function:

```
value.split(",")
```

This would create the array containing the days of the week:

```
["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"]
```

This can be combined with array operations like ‘sort’. For example, assuming the cell contains the same value as above, then the function:

```
value.split(",").sort()
```

Would result in an array containing the days of the week sorted in alphabetical order:

```
[“Friday”, “Monday”, “Saturday”, “Sunday”, “Thursday”, “Tuesday”, “Wednesday”]
```

To output a value from an array you can either select a specific value depending on its position in the list (with the first position treated as ‘zero’). For example:

```
value.split(“,”)[0]
```

Would extract the first value from the array created by the ‘split’ function. In the above example this would be “Monday”

You can also join arrays together to make a ‘String’. The GREL expression would look like:

```
value.split(“,”).sort().join(“,”)
```

Taking the above example again, this would result in a string with the days of the week in alphabetical order, listed with commas between each day.

Regular Expressions

A ‘regular expression’ (sometimes called a ‘regex’) is a way of representing patterns in text strings. These can be used to search for text that matches the pattern represented by the regular expression. Regular expressions typically surrounded by ‘/’ characters.

To write a regular expression you need to know the special syntax used to represent different types of characters that can occur in a text string. The table below introduces some of these:

Type of character	Regular expression syntax	Explanation
Any character	.	A ‘period’ can represent any character at all - essentially a ‘wildcard’
A list or range of characters	[<list/range to be matched>]	You can put a list of characters or a range of characters inside square brackets to match any of the characters in the list. e.g. [ABC] matches A or B or C (note this is case sensitive). [A-Z] matches any uppercase letter [A-Za-z0-9] matches any upper or lower case letter or any digit
Any digit	\d	The syntax ‘\d’ will match any single digit character (equivalent to [0-9])
Any ‘word’ characters	\w	The syntax ‘\w’ will match any character that can be part of a ‘word’. In practice this means any letter (upper or lower case), any digit or the underscore character. (equivalent to [A-Za-z0-9_])
Any ‘whitespace’ characters	\s	The syntax ‘\s’ will match any ‘space’ type character - such as a space, a tab, or a newline.
The start of a string	^	The syntax ‘^’ will match the start of the string - useful if you want to find strings that start with a specific pattern
The end of a string	\$	The syntax ‘\$’ will match the end of the string - useful if you want to find strings that end with a specific pattern

These special characters can be combined with any normal characters to form a regular expression. So to find both the ‘s’ and ‘z’ spellings of ‘organize/organise’) could be:

/organi.e/

Here the ‘.’ character can represent any character and so this would match both ‘organise’ and ‘organize’. Because ‘.’ can represent any character this could also catch other patterns that might occur. To be more specific you could use:

/organi[sz]e/

These (and other) character matches can be combined with ‘repetition’ operators, which allow you to say how many times a character or pattern is repeated. Repetition operators apply to the character or expression immediately preceding the operator. The repetition operations are:

Repetition character	Meaning	Explanation/Example
*	The preceding character/expression can be repeated any number of times (including zero)	The regular expression <code>./*</code> represents any text string at all (any character repeated any number of times)
+	The preceding character/expression can be repeated one or more times	Unlike the ‘*’ using a ‘+’ means the character must appear at least once. The expression <code>/head\s+/rest/</code> would match ‘head rest’ (one space), ‘head rest’ (two spaces) but not ‘headrest’.
?	The preceding character/expression can be repeated 1 or zero times	Essentially makes a character optional in a regular expression. The regular expression <code>/colou?r/</code> would match both ‘colour’ and ‘color’

In addition you can specify exact numbers of repetitions, or a maximum/minimum number of repetitions using curly brackets containing one or two numbers:

`/a{2}/`

Matches the letter ‘a’ appearing twice (i.e. matches ‘aa’)

`/a{2,4}/`

Matches the letter ‘a’ appearing a minimum of two times or a maximum of four times (i.e. matches any of ‘aa’, ‘aaa’, ‘aaaa’)

Going Further

Write a regular expression that would find a four digit number in a string

Write a regular expression that would find a date written using the format ‘dd-MM-yyyy’

Write a regular expression that would find a date written in either the format ‘dd-MM-yyyy’ or ‘dd-MM-yy’

Write a regular expression to find a thirteen digit ISBN

There are many online resources which provide tutorials on using Regular Expressions, including:

- <http://www.regular-expressions.info>
- <http://software-carpentry.org/v4/regexp/index.html>

- <http://www.codeproject.com/Articles/939/An-Introduction-to-Regular-Expressions>
- <http://regex.bastardsbook.com>

Using the ‘match’ transformation with Regular Expressions

Another feature of Regular Expressions is that you can ‘capture’ parts of the matched string to do some further work with. In OpenRefine, this is particularly used with the ‘match’ function. The ‘match’ function allows you to extract particular parts of a string by using a regular expression which captures parts of the matching string. To tell the ‘match’ function which bits of the matched string you want to capture, you simply surround those parts of the regular expression with brackets ().

For example, if you have strings like:

```
pp. 40. G. Bryan & Co: Oxford  
pp. 64. W. Cann: Plymouth  
pp. 92. Heath Cranton: London
```

(representing number of pages, publisher and place of publication). You could use the ‘match’ function as follows

```
value.match(/pp. (\d*).*/)
```

This would find the number following ‘pp’ in each row and put it into an OpenRefine Array - so for the rows above you would get:

```
["40"]  
["64"]  
["92"]
```

In the ‘match’ function the regular expression used has to match the full string, but only the parts of the regular expression in brackets are put into the array. A more complex example with the same example strings might be:

```
value.match(/pp. (\d*)\. (.*)\:(.*)/)
```

This has three capture groups - the page number, the publisher and the place of publication - getting the output:

```
["40", "G. Bryan & Co", "Oxford"]  
[ "64", "W. Cann", "Plymouth"]  
[ "95", "Erskine Macdonald", "London"]
```

Exercise 8: Extracting dates of publication from the ‘Place of Publication’ column

In Exercise 4 above you may have noted that in many cases where the ‘date of publication’ value was blank, there was a date in the ‘place of publication’ column. This exercise is to use what you have learnt about facets, transformations, data types and regular expressions to extract these dates and put them in the ‘Date of Publication’ column.

- Make sure you are only working with records where the Date of Publication is blank [Hint: See Exercise 4]
- Working with the Place of Publication column, use the ‘Add column based on this column’ function [Hint: Look in the drop down menu for this column]
- Use the ‘match’ function with a regular expression to find where the Place of Publication column ends with a four digit number
 - When using the ‘match’ function, you have to use ‘capture groups’ in your regular expression (see above under ‘Regular Expressions’)
 - The output of the ‘match’ function is an Array - you’ll need to get a string value from this

4. Advanced Refine

Looking up data from a URL

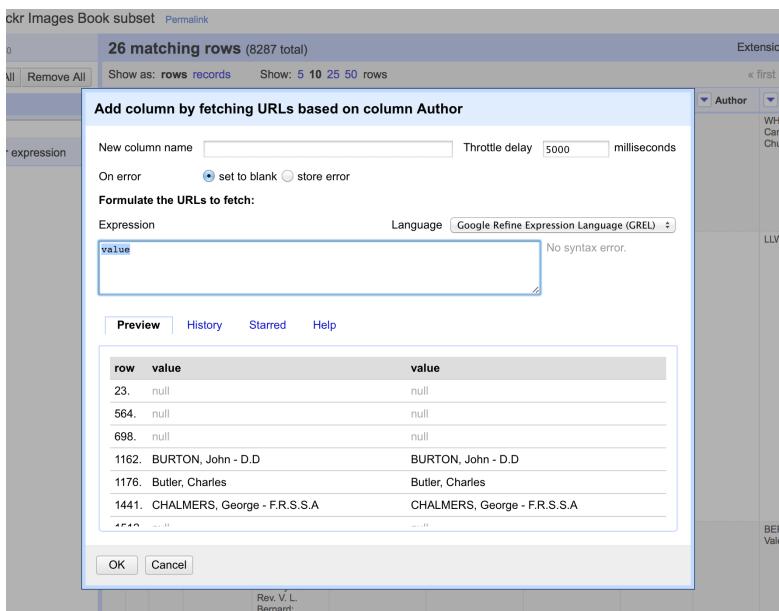
OpenRefine can retrieve data from URLs. This can be used in various ways, including looking up additional information from a remote service, based on information in your OpenRefine data.

As an example, you can look up names against the Virtual International Authority File (VIAF), and retrieve additional information such as dates of birth/death and identifiers.

Typically this is a two step process - firstly a step to retrieve data from a remote service, and secondly to extract the relevant information from the data you have retrieved.

To retrieve data from an external source, from the drop down menu at a column heading use the option 'Edit column' -> 'Add column by fetching URLs'.

This will prompt you for a GREL expression to create a URL. Often this would be a URL that uses existing values in your data to build a query.



For example if you have a list of Author names, and wish to retrieve information from VIAF you can build a URL by appending the name to the end of a general VIAF URL as follows:

```
"http://viaf.org/viaf/AutoSuggest?query="+ escape(value, 'url')
```

(assuming that the column you are working from contains names the 'value' variable will contain the value from the current cell)

This VIAF service returns a list of possible matches in a format called 'JSON'. There is GREL function to extract data from JSON formatted data called 'parseJSON'. This can be used to extract values from the JSON retrieved from VIAF. For example the following expression will extract the first name with a VIAF ID.

```
forEach(value.parseJson().result, v, v.term + ' | ' + v.viafid)[0]
```

The process of retrieving data from external services in this way can be very slow, and is best used on small data sets

Exercise 9: Retrieving VIAF IDs for Authors

- Use the drop down menu at the top of the Date of Publication column and select ‘Text Filter’
- Type ‘1800’ into the Date of Publication text filter - this should reduce the number of records you are working with to about 26
- Use the drop down menu at the top of the Author column and select ‘Edit column’ -> ‘Add column by fetching URLs’
- In the ‘New column name’ box type “VIAF JSON”
- In the ‘Expression’ box, type the expression:
 - “`http://viaf.org/viaf/AutoSuggest?query=`+ escape(value, ‘url’)
- Click ‘OK’
- Wait for OpenRefine to retrieve the data from VIAF - this may take a few minutes
- Use the drop down menu at the top of the new “VIAF JSON” column and select “
- In the ‘Expression’ box, type the expression:
 - `forEach(value.parseJson().result, v, v.term + ' | ' + v.viafid)[0]`

This should leave some of the boxes in the column populated with author names and IDs from VIAF.

Reconciliation services

Reconciliation services allow you to lookup terms from your data in OpenRefine against external services, and use values from the external services in your data.

Reconciliation services can be more sophisticated and quicker than using the method described above to retrieve data from a URL. However, to use the ‘Reconciliation’ function in OpenRefine requires the external resource to support the necessary service for OpenRefine to work with, which means unless the service you wish to use supports such a service you cannot use the ‘Reconciliation’ approach.

Extensions

The functionality in OpenRefine can be enhanced by ‘extensions’ which can be downloaded and installed to add functionality to your OpenRefine installation.

A list of Extensions is given at <https://github.com/OpenRefine/OpenRefine/wiki/Extensions>

One of these extensions tries to work around the limitation of Reconciliation services described above, by making it possible to use a reconciliation service against ‘linked data’¹ sources which have SPARQL endpoints². For more information on this see the ‘RDF Extension’ at <http://refine.deri.ie/> An example of how this works is given in more detail at <http://refine.deri.ie/showcases>.

Records and Rows

All the examples above use OpenRefine in ‘Row’ mode, where it is assumed that each row in the table represents a ‘record’.

¹ Linked Data is a specific data format that is seeing increased usage in the library and cultural heritage sector

² SPARQL is a language used to query ‘Linked Data’ and a “SPARQL Endpoint” is the URL to which such queries can be sent

However, OpenRefine supports a more complex model of ‘Records’ which allows you to have multiple values for a single column in a single record. For example, it is not unusual for a book to have multiple people (or ‘contributors’) involved in the creation of the book.

In the screenshot below you can see the options to select a to ‘Show as’ rows or records, and also see how row 11 has two contributors listed.

8287 records										Extensions: Utility
Show as: rows records Show: 5 10 25 50 records										« first < previous
All	Identifier	Title	Edition Statement	Place of Publication	Date of Publication	Publisher	Author	Contributors	Corporate Author	
		preface signed: T. A.]								
11.	000001280	An Account of the many and great Loans, Benefactions and Charities, belonging to the City of Coventry ... A new edition. [The dedication signed: AB, CD, EF, GH, &c. By Edward Jackson and Samuel Carte.]		Coventry	1802	Printed by J. Turner	edit	CARTE, Samuel.		
								JACKSON, Edward - Rector of Southam, and CARTE (Samuel)		
12.	000001808	Erindringer som Bidrag til Norges Historie fra 1800-1815. Anden Udgave ... Udgivet med nogle Rettelser og Tillæg af Christian C. A. Lange. Med Forfatterens Portraet, og hans Biographi af Amtmand J. C. Aall		Christiania	1859		AALL, Jacob.	AALL, J. C.		
								LANGE, Christian Christoph Andreas.		

The “records” approach is occasionally useful and can be used when you have two rows in the original data set that represent the same ‘thing’ (book/person/place/etc.) and wish to merge the two rows into a single row, preserving information from both original rows.

For more information on how you can create Records in OpenRefine see <http://googlerefine.blogspot.co.uk/2012/06/create-records-in-google-refine.html>

Using the ‘cross’ function to lookup data in other OpenRefine projects

As well as looking up data in external systems using the methods described above, it is also possible to look up data in other OpenRefine projects on the same computer. This is done using the ‘cross’ function.

The ‘cross’ function takes a value from the OpenRefine project you are working on, and looks for that value in a column in another OpenRefine project. If it finds one or more matching rows in the second OpenRefine project, it returns an array containing the rows that it has matched.

As it returns the whole row for each match, you can use a transformation to extract the values from any of the columns in the

You can use to compare the contents of two OpenRefine projects, or to use data between the two projects.