

Validating and Cleaning Data

This exercise focusses on using tools to validate, clean explore data sets.

Introduction

A big problem with publicly available datasets is the number of errors within them. These problems vary from simple spelling errors, to the more complex problems involving misuse of units. This exercise is going to evaluate the following problems and related solutions:

1. **Date Validation**

One of the most common problems in data is mixed date formats, this can be particularly troublesome when you have British and American date formats e.g. (7/12/2012 and 12/31/2012).

2. **Multiple Representations**

Most common in datasets containing abbreviations, for example in location data or role based data. It is common that abbreviations will change and even be present in fully expanded form (e.g Vice-President Marketing and VP Marketing)

3. **Duplicate Record Detection**

Duplicated records are common place both at the point of entry (by a human) but also a common occurrence when exporting a huge amount of data from multiple systems. It is often the case that the data has been duplicated in order to speed up searching across multiple domains where the data is applicable in both.

4. **Summation Records**

When data has been extracted from a spreadsheet application, it is common to be left with both columns and rows of data containing the sums (or other formula) of the other data. While not an error, it is inconvenient when you want to re-process the data.

5. **Mixed use of numerical scales**

A common, but critical, failure in data that can lead to audit failure. Outliers are often clear to see as one record may contain a figure multiple factors bigger than any other.

6. **Redundant Data**

Redundant data is not required, thus it is common that errors are made when entering it.

7. **Numeric Ranges**

Numeric ranges, often used to anonymise data, cause problems when wanting to explore and visualise the data.

8. **Spelling Errors**

Last but not least, while not critical in all cases, spelling errors can lead to awkwardness when querying and visualising data.

Required datasets

In order to carry out this exercise three datasets are required. Although the datasets are genuine, they have all been modified for this exercise. The modified datasets are available from the course website.

Download each of the datasets onto your laptop for later use. Do not open them in any application including excel, this may result in you changing the dataset if it is re-saved.

Dataset 1 – Louisiana Secretary of State Officials

Download: <http://training.theodi.org/resources/dataset1.xls>

This dataset lists the statewide and multi-parish elected officials, all elected officials in a parish, and all elected officials in an office e.g. all sheriffs in the state of Louisiana.

The original dataset is available at: <http://www.sos.la.gov/tabid/136/Default.aspx> (removed as of 1/7/2013)

Dataset 2 – Projects Dataset

Download: <http://training.theodi.org/resources/dataset2.csv>

This dataset lists project data available from the US Governments IT Dashboard system at http://www.itdashboard.gov/data_feeds. It covers the projected and actual costs and timings of a number of government funded projects in the US.

Dataset 3 – UK GP Earnings

Download: <http://training.theodi.org/resources/dataset3.xls>

This dataset lists earnings data for medical doctors in the UK from 2009. The original dataset is available from <http://data.gov.uk/dataset/gp-earnings-and-expenses-2009-10>

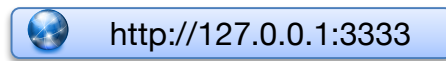


Importing into Google/Open Refine

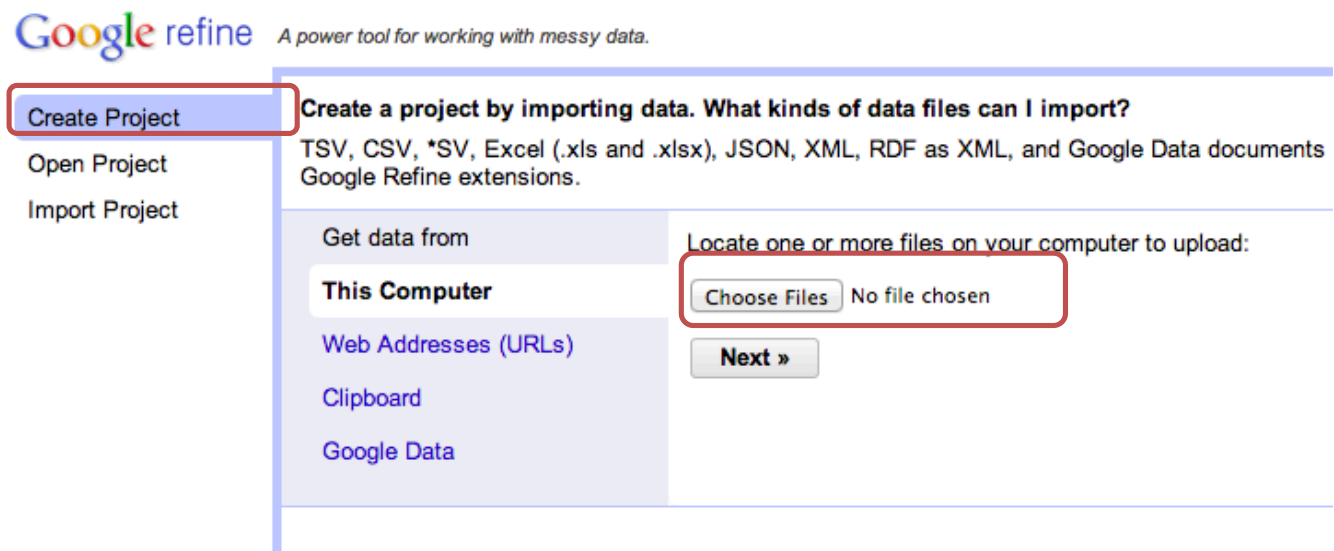
In order to process the data requires the Google Refine (soon to be Open Refine) tool available from openrefine.org.

Refine is an application that runs on your local machine, meaning that you don't have to upload a large dataset to a web service. Additionally this has the benefit that the data remains private.

Once installed and running it should open a browser window on the refine home screen.



From the home screen, create a new project by choosing a downloaded dataset on your machine.



Google refine *A power tool for working with messy data.*

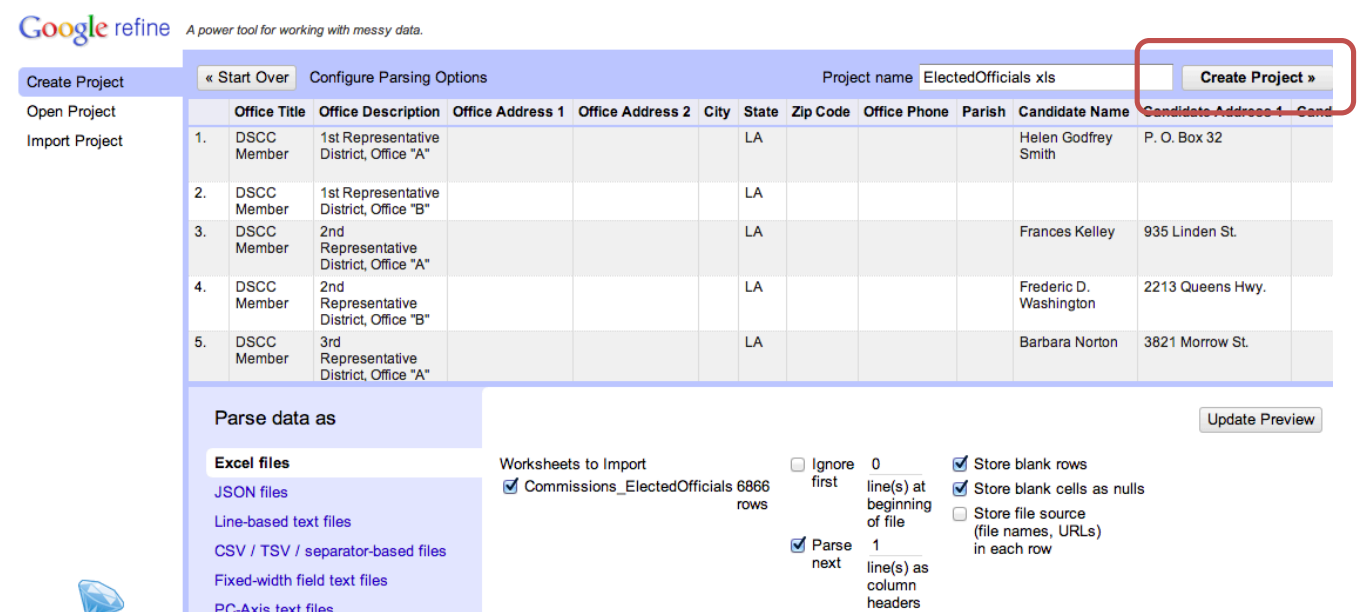
Create Project
Open Project
Import Project

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
Google Refine extensions.

Get data from
This Computer
Web Addresses (URLs)
Clipboard
Google Data

Locate one or more files on your computer to upload:
Choose Files No file chosen
Next »

The following screen is the **import** screen and shows you a preview of what Refine thinks the dataset should look like. Check the titles and columns look correct before clicking **create project**. If the import options look wrong you can adjust them using the panel at the bottom on the screen.



Google refine *A power tool for working with messy data.*

Create Project
Open Project
Import Project

« Start Over Configure Parsing Options Project name ElectedOfficials.xls Create Project »

	Office Title	Office Description	Office Address 1	Office Address 2	City	State	Zip Code	Office Phone	Parish	Candidate Name	Candidate Address 1	Candidate Address 2
1.	DSCC Member	1st Representative District, Office "A"				LA				Helen Godfrey Smith	P. O. Box 32	
2.	DSCC Member	1st Representative District, Office "B"				LA						
3.	DSCC Member	2nd Representative District, Office "A"				LA				Frances Kelley	935 Linden St.	
4.	DSCC Member	2nd Representative District, Office "B"				LA				Frederic D. Washington	2213 Queens Hwy.	
5.	DSCC Member	3rd Representative District, Office "A"				LA				Barbara Norton	3821 Morrow St.	

Parse data as
Excel files
JSON files
Line-based text files
CSV / TSV / separator-based files
Fixed-width field text files
PC-Axis text files

Worksheets to Import
☒ Commissions_ElectedOfficials 6866 rows

☐ Ignore first 0 line(s) at beginning of file
☒ Parse next 1 line(s) as column headers

☒ Store blank rows
☒ Store blank cells as nulls
☐ Store file source (file names, URLs) in each row

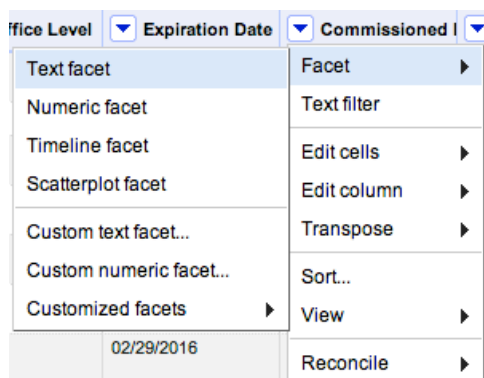
Update Preview



1. Date Validation (Dataset 1)

One of the most common problems in datasets is that of mixed date formats. Sometimes the mix is simple to spot, e.g. 8-Sep-2013 vs 8/9/2013, sometimes not, e.g. 8/9/2013 vs 9/8/2013.

Due to this problem, the majority of tools, including refine, will simply import the data as a string object and not worry about the format or content of the object. In the case of refine a string object is known as a text object and can be browsed using a **text facet**.



In the dataset we are looking at in this exercise we are going to look at the range of dates in the **commissioned date** column.

To apply a text facet, click the **downward arrow** next to the column title and select **text facet**.

Doing this will bring up a facet browser that you can use to view all the data in this column groups together. A quick scroll through

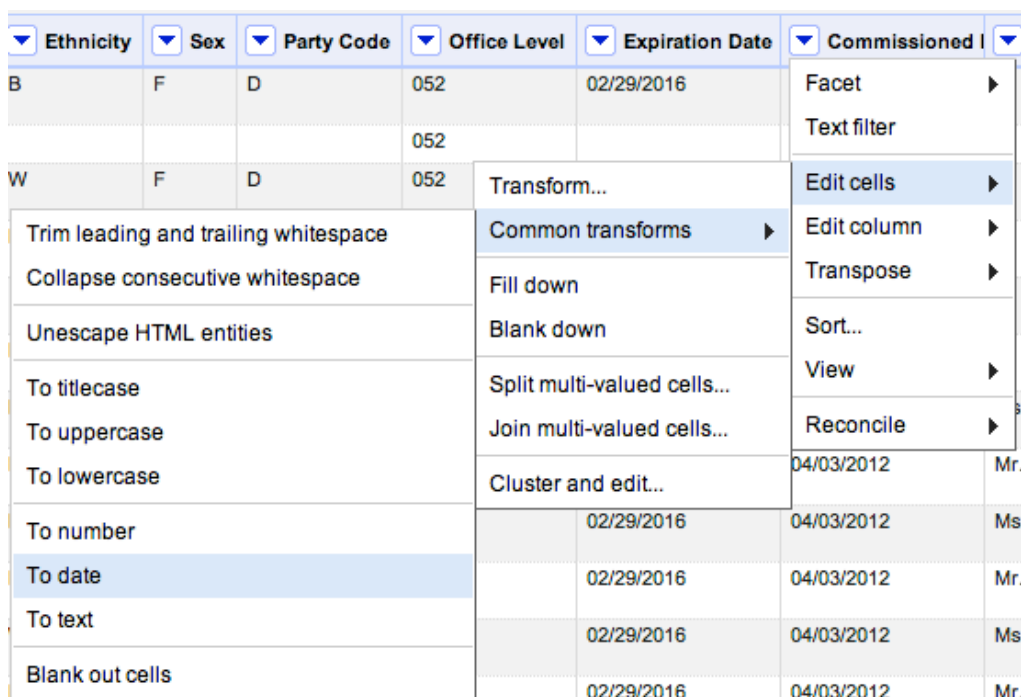
this panel will reveal that we are in an American date format, with month first. There is one invalid date affecting 17 records.

To change this value, we could hover over the value and click the edit link, however we are going to look at a different method using a cell transform.

A cell transform allows us to change the type of the object. In this case from text to a date object.



To do this select the **to date** option from the **column transforms** menu as shown.

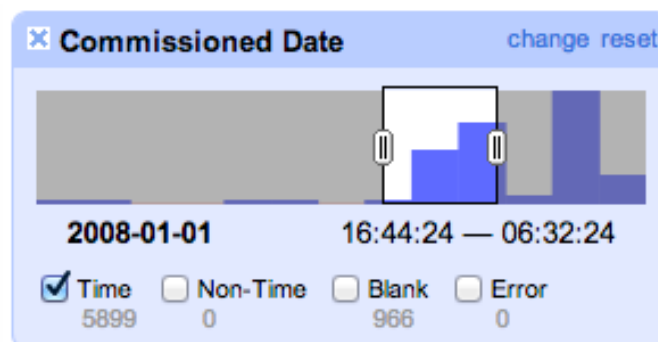


Once done you will see that the **text facet** we applied previously is now full of random values that make no sense. This is because we cannot apply a text facet to an object that is not actually text anymore. At this point remove the text fact by clicking the close button in the top left corner of the facet browser.



In translating our text to a date object, refine has parsed over all the values in the date column and attempted to match the most common date format and used this as the basis to correct errors. In this case it would have recognised that the UK date format was being used and automatically corrected our 17 records from before. As this is such a common error, libraries for recognizing date formats are commonplace and used a lot on the web as well as other platforms.

In order to explore our new date objects we can apply a **timeline facet**. To do this click the downward arrow next to the column title and select **timeline facet** from the facet sub menu. The facet that appears will now display the range of dates and the number of items as a graph.



You probably want to untick the blank box while browsing the data in this way so that the facet is only displaying records that have a date associated with them. Note that if any of the rows had failed in the date translation they would appear as errors in this facet and would required manual investigation to clean.



2. Multiple Representations (Dataset 1)

Due to the unique ways that people like to save time in data entry by abbreviating everything, it is very common to end up with several different representations of the same thing.

Thankfully the advanced clustering features of **Refine** can help us out.

In this example we are going to use our Louisiana dataset and apply a **text facet** to the *Office Title* column. In doing this we can immediately see many errors in the data.

The errors highlighted all seem to involve trailing spaces and we can correct this in two ways. Firstly we can directly edit each value by hand, by hovering over it and clicking the **edit** button. Try this with the **Assessor** values, upon saving your edit you should see them group together showing 64 records.

Perhaps a more useful way however is to use a **trim spaces transform** on the *Office Title* column to clean them all in one go.

While this has eliminated many of the errors, others still remain, such as “Council Member” and “CouncilMember”. To fix these errors we can use the clustering techniques available in Refine. To access these press the **cluster** button from the facet browser.

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
5	375	<ul style="list-style-type: none">Council Member (367 rows)Councilmember (5 rows)Council Member I (1 rows)Council Member II (1 rows)Council Member III (1 rows)	<input type="checkbox"/>	Council Member
2	801	<ul style="list-style-type: none">Alderman (799 rows)Aldermen (2 rows)	<input type="checkbox"/>	Alderman
2	284	<ul style="list-style-type: none">Councilman (273 rows)Councilmen (11 rows)	<input type="checkbox"/>	Councilman
2	17	<ul style="list-style-type: none">Council Member at Large (15 rows)Councilmember at Large (2 rows)	<input type="checkbox"/>	Council Member at Large

At the top of the clustering screen you can pick from many scientific methods and keying functions which all cluster data in slightly different ways. The method that will work best will very much depend upon your dataset and thus it is worth browser through each method and function to find which one best suits your needs.

It may be necessary to use a combination of methods and functions, each time selecting a number of records you want to **merge**, entering the **new value** and then pressing **merge selected and re-cluster**.



3. Duplicate Record Detection (Dataset 1)

In order to identify duplicate rows we are going to look at the data in the *Candidate Name* column. Once again we are going to use the **clustering** function, but this time we need to examine the data more closely.

To bring up the clustering panel, select **cluster and edit** from the **edit cells** menu from the dropdown of the *Candidate Name* column.

As in the multiple representations section, it is recommended that you look at the multiple functions to find that which best shows likely duplicate records.

Candidate Name	Candidate Addr	Candidate
Facet	O. Box 32	
Text filter		
Edit cells	Transform...	
Edit column	Common transforms	
Transpose	Fill down	
Sort...	Blank down	
View	Split multi-valued cells...	
Reconcile	Join multi-valued cells...	
Larry Ferdinand	34	Cluster and edit...

Unlike in the previous exercises we do not want to *change* values, we want to *remove* duplicates. To do this we first need to confirm that the data is duplicated. To discover this, hover your pointer over a cluster and then select the **Browse this Cluster** option.

Cluster & Edit column "Candidate Name"

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 "Gödel" and "Godel" probably refer to the same person. [Find out more ...](#)

Method: key collision Keying Function: fingerprint 5 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
2	2	<ul style="list-style-type: none">"Tony" Guillory (1 rows)Tony Guillory (1 rows)	<input type="checkbox"/>	Tony Guillory
2	2	<ul style="list-style-type: none">Kenneth O. Stinson (1 rows)Kenneth O. Stinson (1 rows)	<input type="checkbox"/>	Kenneth O. Stinson
2	2	<ul style="list-style-type: none">Frank John LaBruzzo (1 rows)John Frank LaBruzzo (1 rows)	<input type="checkbox"/>	Frank John LaBruzzo
2	2	<ul style="list-style-type: none">"Tony" Jurich (1 rows)Tony Jurich (1 rows)	<input type="checkbox"/>	Tony Jurich
2	3	<ul style="list-style-type: none">Russell P. Pavich (2 rows)Russell P. Pavich (1 rows)	<input type="checkbox"/>	Russell P. Pavich

[Browse this cluster](#)

Rows in Cluster

Average Length of Choices

Length Variance of Choices

Select All Deselect All Merge Selected & Re-Cluster Merge Selected & Close Close

Using the new window that pops up, we can then browse just that cluster and **star** any duplicated data that we wish to later remove. Once done, close the window or tab to return to the original dataset.

2 matching rows (6865 total)										
Show as: rows records Show: 5 10 25 50 rows										
All	Office Title	Office Descriptio	Office Address	Office Address	City	State	Zip Code	Office Phone	Parish	Candidate Name
	1955.	Mayor	Town of Vinton	1200 Horridge St.	Vinton	LA	70668	337-589-7453	CALCASIEU	Kenneth O. Stinson
	1956.	Mayor	Town of Vinton	1200 Horridge St.	Vinton	LA	70668	337-589-7453	CALCASIEU	Kenneth O. Stinson

Do this for a number of duplicated records before closing the clustering screen.

To view all the rows you starred apply a **star facet** to the *All* column, select the true values and then delete them by selecting **Remove all matching rows** from the **edit rows** menu. If you cannot see any starred rows, ensure that you don't have any facets currently applied.

All	Office Title	Office Descriptio	Of
Facet		Town of Vinton	1200 H
Edit rows	Star rows		
Edit columns	Unstar rows		
View	Flag rows		
	Unflag rows		
	Remove all matching rows		



4. Summation Records (Dataset 2)

It is often the case that data exported from a spreadsheet application will contain summation rows and columns. While the columns are easier to spot, the rows are much harder in a large dataset. A little tip is to browse right to the end of the dataset in order to see what the very last record is. This can be done in **Refine** by clicking the *last* button.

2494 rows							
Extensions: Freebase ▼							
Show as: rows records Show: 5 10 25 50 rows « first < previous 2451 - 2494 next > last »							
▼ All	▼ Unique Investment	▼ Business Case I	▼ Agency Code	▼ Agency Name	▼ Investment Title		
☆	🗨	2490.	429-000001001		1035	429	Nuclear Regulatory Commission
☆	🗨	2491.	429-000001001		1035	429	Nuclear Regulatory Commission
☆	🗨	2492.	429-000001001		1035	429	Nuclear Regulatory Commission
☆	🗨	2493.	429-000001001		1035	429	Nuclear Regulatory Commission
☆	🗨	2494.	Total				

Let's start by starring the "Total" row for later removal. Now we know that they exist, we should check to see if there are any more rows and try to find what they represent.

▼ Unique Investment	▼ Business Case I	▼ Agency C
Facet	Text facet	
Text filter	Numeric facet	

Apply a **text facet** using the drop down next to the *Unique Investment* column.

Scroll to the bottom of the facet and select all "Total" rows and **star** these. Starring rows is done from the dropdown menu of the **All** column and is available under **edit rows**.

Unique Investment Identifier		change invert reset
600 choices	Sort by: name count	Cluster
429-000001020	9	
429-000001100	4	
429-000002005	4	
429-000002016	7	
429-000002080	16	edit include
Total	26	exclude
Facet by choice counts		

While we are in the facet also note the row numbers where the total exists. As there are many of them, we might conclude that this one dataset is an export of many worksheets. Clearing the facet (done by closing it on the left) and browsing to one of the recorded row numbers allows us to gain an idea about how the data was represented in the various worksheets.

☆	🗨	172.	005-000002376		1099	5	Department of Agriculture	Conservation Delivery Streamline Initiative (CDSI)
☆	🗨	173.	Total					
☆	🗨	174.	006-000525200	edit	629	6	Department of Commerce	BEA Estimation Information

From the data displayed it looks like the totals are per agency. When happy that the summations are understood, delete the total rows such that they don't spoil the later processing. To do this apply a **Facet by star** from the **All** column, select **true** in the facet. Finally from the **All** dropdown select **remove all matching rows** from the **edit rows** menu.

▼ All	▼ Unique Investment	▼ Business Case I
Facet		
Edit rows	Star rows	
Edit columns	Unstar rows	
View	Flag rows	
☆	925.	Total
☆	1052.	Total
☆	1124.	Total

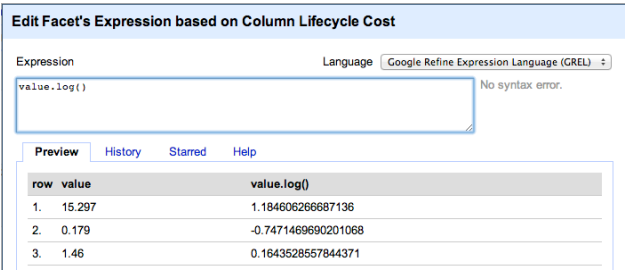
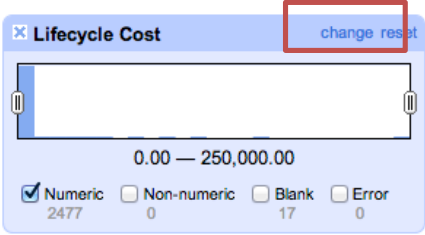


5. Mixed use of numerical scales (Dataset 2)

With the projects dataset being all about costing and budgets, we should probably take a look at the numerical data in these columns to see if there is consistent usage of units.

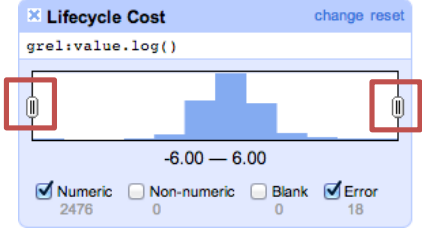
Applying a **numerical facet** to the *Lifecycle Cost* column is useful in some ways, but doesn't truly represent the distribution of values from a norm. What we can see here is a power curve; there are lots of low cost projects and a long tail of few high cost projects.

In order to analyse this distribution we can use the mathematical log function to examine the power curve distribution on an exponential scale. To do, click the change button. From the box that appears we can apply filters and programmatic changes to the values in the columns.



In order to more clearly display the distribution of our values we are going to change the values so we can view them on a log scale. This can be done by adding `.log()` to the end of our value.

Using this distribution we can now see a normal distribution of values. Further we can much more easily analyse those extreme values at each end of the scale. In order to examine the extremes, drag the two controls in from either side as appropriate to filter the data.



By looking at this data, as well as the column titles of other columns, it should be relatively clear that the units of this column are probably \$M. There are many low cost projects, however there is also one 14 month project with a huge cost.

Project	Projected/Actual	Lifecycle Cost	Schedule Variance	Schedule Variance (%)	Cost Variance (\$)	Cost Variance (%)	Planned Cost (\$)	Projected/Actual
		119098.612	0	0	-3.119485	-2.62	119.099001	122.218486

Looking at the difference between lifecycle cost and planned cost should reveal the extent of the problem and allow it to be fixed.

Imagine the knock on effect this had with the totals!

N.B. While the totals rows were added to the data for the purposes of this exercise. The mistaken project cost of 117098 million existed in the original dataset!



6. Redundant Data (Dataset 2)

During the summation records exercise it was discovered that the data appears to be grouped by Agency.

☆	🗨	171.	005-000002376		1099		5	Department of Agriculture	Conservation Delivery Streamline Initiative (CDSI)
☆	🗨	172.	005-000002376		1099		5	Department of Agriculture	Conservation Delivery Streamline Initiative (CDSI)
★	🗨	173.	Total						
☆	🗨	174.	006-000525200	edit	629		6	Department of Commerce	BEA Estimation Information Technology System (BEA-EITS)
☆	🗨	175.	006-000525200		629		6	Department of Commerce	BEA Estimation Information Technology System (BEA-EITS)

Looking at this data again, it should also be clear to see that we have an *Agency Code* and *Agency Name* columns. While it shouldn't matter that we have both pieces of data, redundant data can also lead to errors. Beneficially, redundant data can often be easier to fix; the more data you have, the clearer the fix is likely to be.

In this exercise we are going to check that the agency codes always match the name. In order to do this we are going to amalgamate the data in a single column and then apply a text facet.

Agency Name Investment Title Project ID Agency Pro

Facet

Text filter

Edit cells

Edit column

Transpose

Split into several columns...

Add column based on this column...

From the *Agency Name* column select **add column based on this column** from the **edit column** menu.

This will pop up an expression editing box similar to the one we used in the numerical scales exercise. The default expression simply copies the data from this column to a new one. We are going to change this to copy the data from two columns into a new *Combined Data* column.

Add column based on column Agency Name

New column name: Combined Data

On error: ☒ set to blank ☐ store error ☐ copy value from original column

Expression: value + " (" + cells["Agency Code"].value + ")"

Language: Google Refine Expression Language (GREL)

No syntax error.

Preview History Starred Help

row	value	value + " (" + cells["Agency Code"].value + ")"
151.	Department of Agriculture	Department of Agriculture (5)
152.	Department of Agriculture	Department of Agriculture (5)
153.	Department of Agriculture	Department of Agriculture (5)

Once done, try applying a **text facet** to our new column to find and correct any errors that exist in the dataset.

As an interesting experiment, you could also choose to bring back the total columns and see if the totals correlated to one or more of your fixes.



7. Numerical Ranges – Dataset 3 (Advanced)

In anonymised data it is very common to split numerical data into ranges. However this can make processing and visualising the data a much bigger challenge. In the example below we can see both age range data (e.g. 25-30) and salary data (e.g. >25k).

GP_Type	Contract_Type	Country	Gender	Age_Band	Estimated_Popu	Effective_Return	Average_Gross_
Salaried	GPMS	UK	Male	20-35	1100	700	>20k<30k
Salaried	GPMS	UK	Male	35-40	550	350	>30k
Salaried	GPMS	UK	Male	40-50	300	150	>10k<20k
Salaried	GPMS	UK	Male	50-65	250	100	>10k<20k

By applying a **text facet** to *Gender* and at the same time a **numeric facet** to *Average Gross Earnings from Employment*, you should be able to see that (in this dataset), men are earning more than women. Note also the character encoding error on the column titles, meaning the column titles give no indication of units.

In order to explore this further it would also be good to apply a **numeric facet** to *Age Band* and *Average Gross Earnings from Self Employment*, however the data in these columns is not numeric. We could try using the **to number** function under **common transforms**, however this does not work on this data so some other method needs to be applied. In this example we use the **expression editor** and the **python** language to do some processing on the values.

To bring up the expression editor, choose **custom numeric facet** from the *Age Group* column.

In both this and the next example, the choice has been made to remove the ranges and simply change these into numeric values that represent the mid point (as a whole number).

Custom Numeric Facet on columnAge_Band

Expression

```
bits = value.split("-");  
diff = int(bits[1]) - int(bits[0]);  
diff = diff / 2;  
value = int(bits[0]) + diff;  
  
return value;
```

Language python

No syntax error.

Preview

History

Starred

Help

row	value	bits = value.split("-"); diff = int(bits[1]) - int(bits[0]); diff = diff / 2; value = int(bits[0]) + diff; return value;
5.	20-35	27
6.	35-40	37
7.	40-50	45

To process the salary data is a little more complicated as we have lots of variations that need to be dealt with. Below is a piece of sample code to process the salary data.

```
value = value.replace('k','000');  
  
if value[:1] == ">":  
    value = value[1:];  
if value[:1] == "<":  
    value = value[1:];  
if value[:1] == "=":  
    value = value[1:];  
  
bits = value.split("<");  
if len(bits) < 2:  
    return int(value);  
  
diff = int(bits[1]) - int(bits[0]);  
diff = diff / 2;  
value = int(bits[0]) + diff;  
return int(value);
```