Validating and Cleaning Data

This exercise focusses on using tools to validate, clean up and perform some initial exploration over various data sets. Due to the current state of the tools, we are going to use of number of different tools to show the principals. It is hoped that in the future a more complete tool will become available for this purpose.

Introduction

A big problem with publically available datasets is the number of problems with them. These problems vary in type 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.

3. **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.

4. **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 facetted editing.

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 have a budget column 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.

Importing Data

In order to carry out this exercise three datasets are required. Although the datasets are genuine, they have been modified for this exercise.

Dataset 1 – Louisiana Secretary of State Officials

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

The modified dataset is available from the course website and needs to be uploaded into **Google Spreadsheets** as well as loaded into **Google Refine**.

Dataset 2 – Projects Dataset

This dataset lists project data available from the US Governments IT Dashboard system at http://www.itdashboard.gov/data_feeds.

This dataset has also been modified and needs to only be loaded into a separate project in **Google Refine**.

Dataset 3 – UK GP Earnings

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

This dataset has been modified and needs to only be loaded into a separate project in **Google Refine**.

Importing into Google Spreadsheets

http://drive.google.com

You will need to login to the URL above with a valid Google account and then upload the dataset using the upload button.

Please ensure the conversion option is enable. This way the uploaded file is transformed into a google spreadsheet. **Note**: This process may take some time.

Importing into Google/Open Refine

Refine is an application that runs on your local machine, once installed and running it should open a browser window on the refine home screen.

http://127.0.0.1:3333

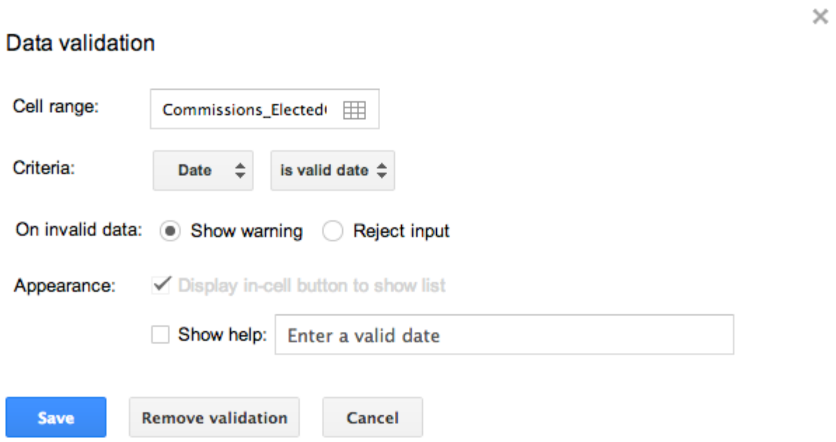
From the home screen, create a new project (per dataset) and run through the import options. In the majority of cases the default selections are correct.

Date Validation (Dataset 1)

Once you have the Louisiana data loaded into Google Spreadsheets, we are going to apply a data validator to the *Expiration Date* and *Commissioned Date* columns to try and locate any errors.

This can be done by selecting both columns and then selecting **Validatation** from the **Data** menu.

This will pop up a validation box in which we need to set criteria that validates date:



Data validation

Cell range: Commissions_Elected

Criteria: Date | is valid date

On invalid data: ● Show warning ○ Reject input

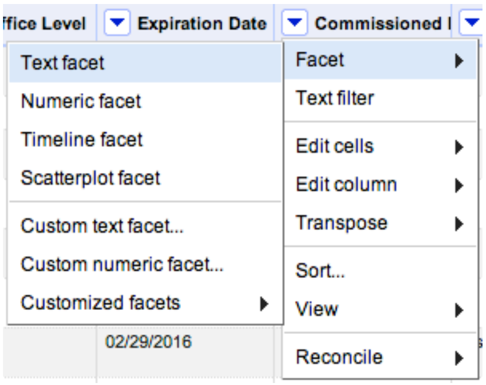
Appearance: ✓ Display in-cell button to show list

□ Show help: Enter a valid date

Google spreadsheets will attempt to guess the correct format and then mark any that don't match this format for you to correct. Note that this operation may take some time to complete. Once done invalid dates will be marked by a red triangle in the corner of the cell.

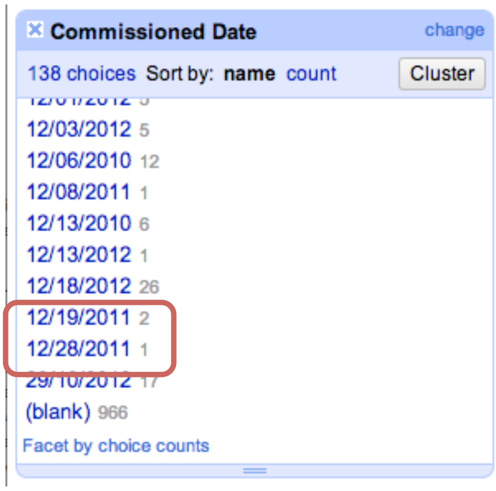
Even when done it may still be hard to see the errors in the dataset and understand which the correct date format is. This is where **Refine** can help.

With the same dataset loaded into refine, we can apply a **text facet** to the *Commissioned Date* column in order to see the range of values.



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.



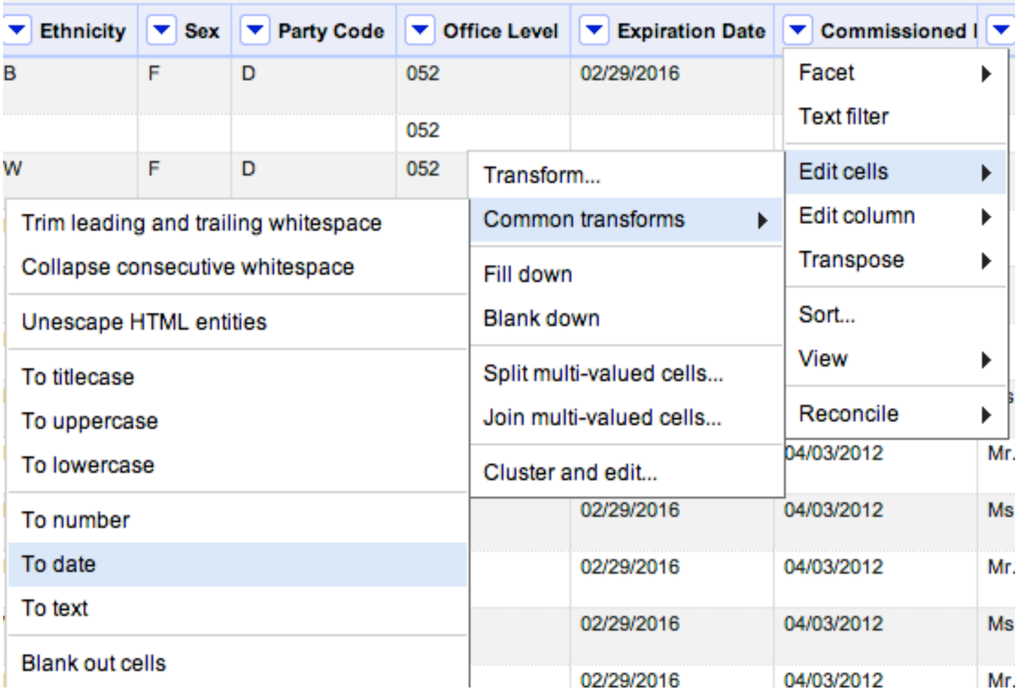
Commissioned Date

138 choices Sort by: **name** count

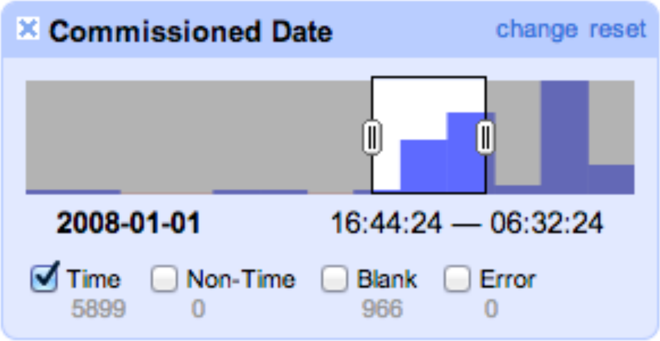
12/01/2012 3
12/03/2012 5
12/06/2010 12
12/08/2011 1
12/13/2010 6
12/13/2012 1
12/18/2012 26
12/19/2011 2
12/28/2011 1
29/10/2012 17
(blank) 966

We can now go back to Google Spreadsheets and use the **find and replace** tool in the **edit** menu to change our values.

Switching back to **Refine**, there is another way to fix bad dates, and that is to apply a cell transform. TO do this select the **to date** option from the **column transforms** menu as shown.



Once done we can then apply a **timeline facet** to the data enabling us to browse the contents of the collection based upon selecting a range of dates.

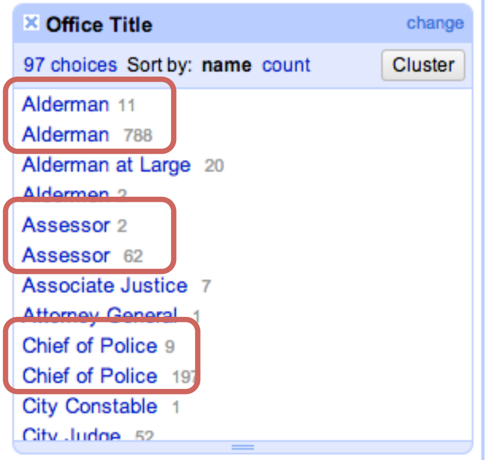


Note: You probably want to untick the blank box while browsing the data in this way.

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.

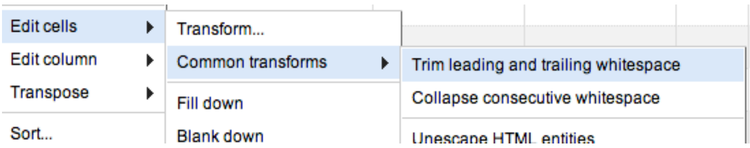
Thankfully the advanced clustering features of **Refine** can help us out.



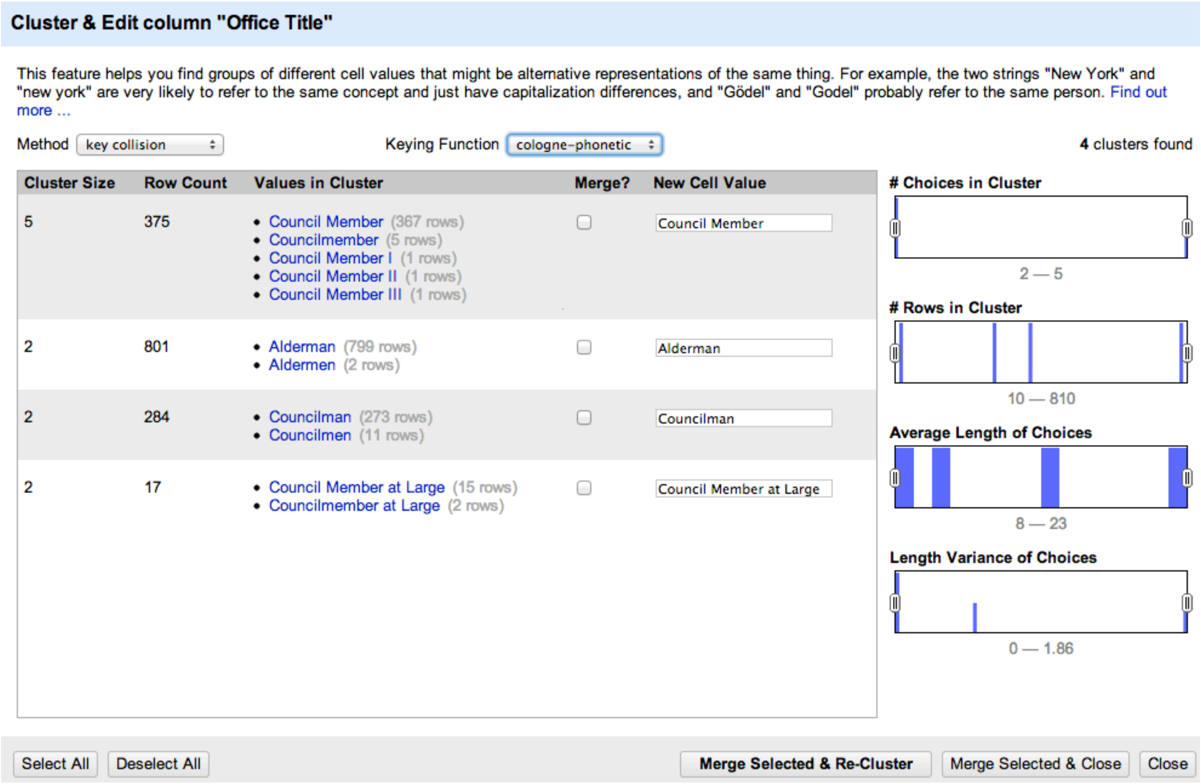
Alderman	11
Alderman	788
Alderman at Large	20
Aldermen	2
Assessor	2
Assessor	62
Associate Justice	7
Attorney General	1
Chief of Police	9
Chief of Police	19
City Constable	1
City Judge	52

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. Perhaps a more useful way however is to use a **trim spaces transform** on the *Office Title* column.



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 **bluster** button from the facet browser.


Cluster & Edit column "Office Title"

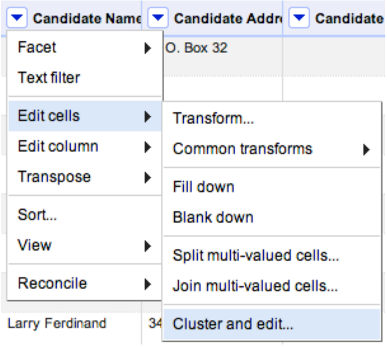
This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer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

Method: key collision   Keying Function: cologne-phonetic   4 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
5	375	• Council Member (367 rows) • Councilmember (5 rows) • Council Member I (1 rows) • Council Member II (1 rows) • Council Member III (1 rows)	□	Council Member
2	801	• Alderman (799 rows) • Aldermen (2 rows)	□	Alderman
2	284	• Councilman (273 rows) • Councilmen (11 rows)	□	Councilman
2	17	• Council Member at Large (15 rows) • Councilmember at Large (2 rows)	□	Council Member at Large

From this window you may need to change the **method** and **function** type in order to best match your requirements. Additionally it might be worth going through the clustering process several times in order to fully clean the data.

Duplicate Record Detection (Dataset 1)

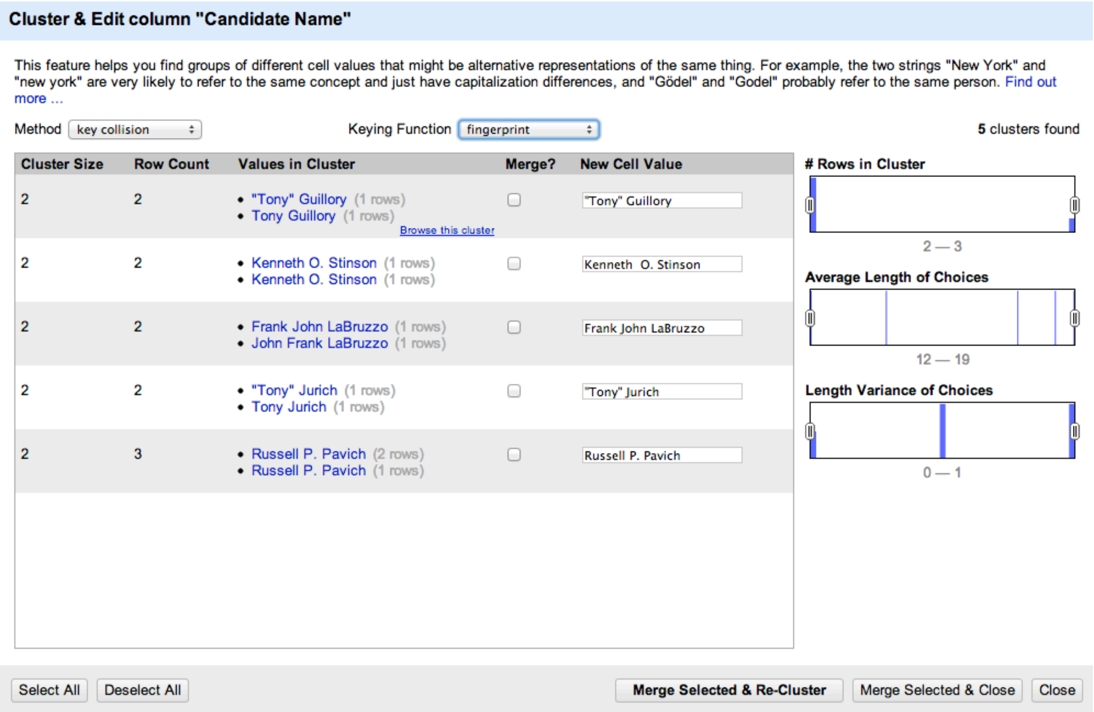


In order to identify duplicate rows we are going to look at the data in the *Candidate Name* column. Once again we are going to use the **clustering** function, but this time we need to examine the data more closely.

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

Unlike in the last exercises we do not want to change values, we want to remove duplicates. First however we 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.



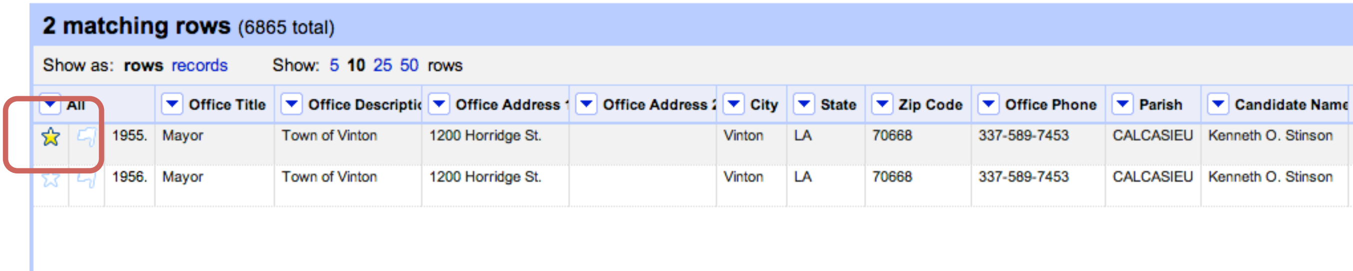
Cluster & Edit column "Candidate Name"

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	• "Tony" Guillory (1 rows) • Tony Guillory (1 rows) Browse this cluster	□	"Tony" Guillory
2	2	• Kenneth O. Stinson (1 rows) • Kenneth O. Stinson (1 rows)	□	Kenneth O. Stinson
2	2	• Frank John LaBruzzo (1 rows) • John Frank LaBruzzo (1 rows)	□	Frank John LaBruzzo
2	2	• "Tony" Jurich (1 rows) • Tony Jurich (1 rows)	□	"Tony" Jurich
2	3	• Russell P. Pavich (2 rows) • Russell P. Pavich (1 rows)	□	Russell P. Pavich

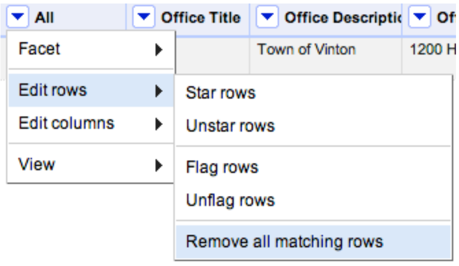
Using the new window that pops up, we can then browse just that cluster and **star** any data that we wish to later remove.



2 matching rows (6865 total)

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

All		Office Title	Office Descriptio	Office Address	Office Address 2	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

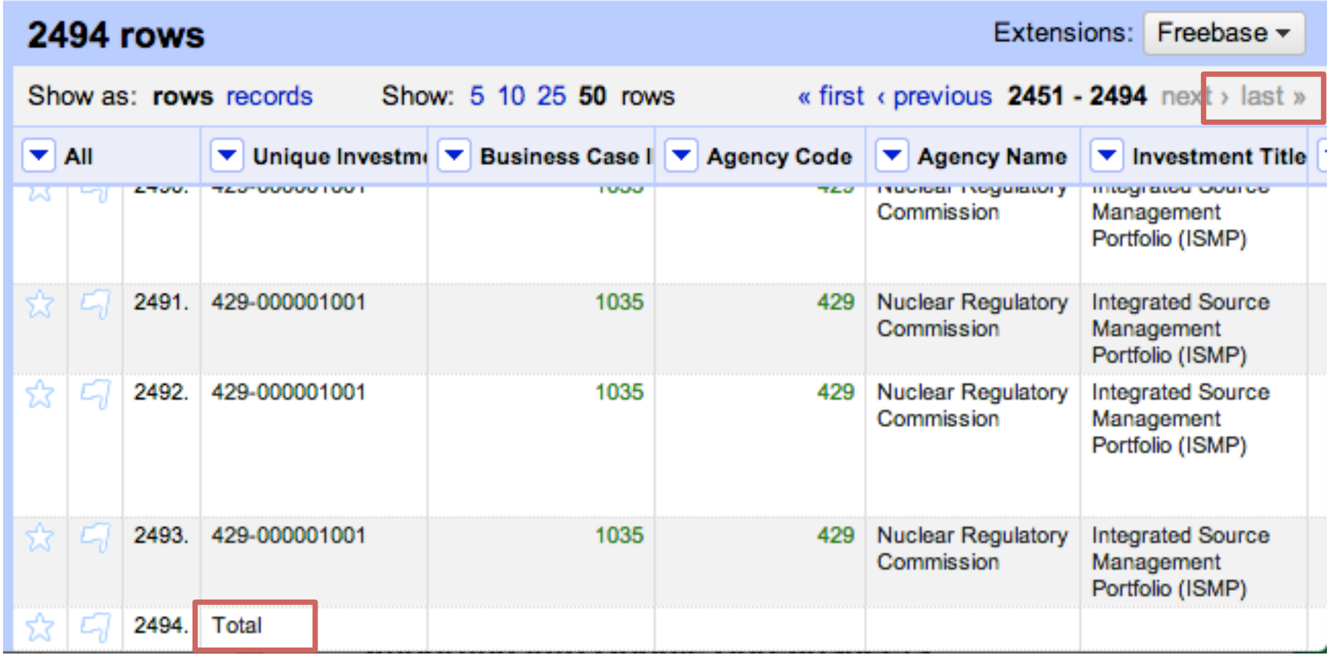


Once finished, close the clustering screen and ensure you can see all rows. To view all the rows you stared apply a **star facet** to the *All* column, select the true values and then delete them by selecting **Remove all matching rows** from the **edit rows** menu.

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.



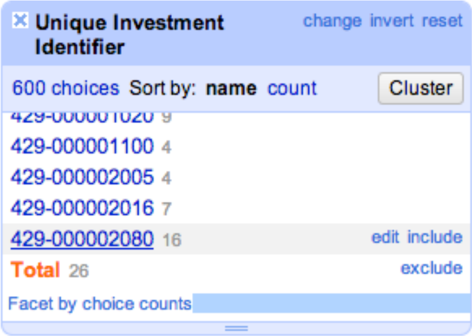
2494 rows Extensions: Freebase ▾

Show as: **rows** records Show: 5 10 25 **50** rows « first ‹ previous **2451 - 2494** next › last »

	All	Unique Investm	Business Case I	Agency Code	Agency Name	Investment Title
☆	2490.				Commission	Management Portfolio (ISMP)
☆	2491.	429-000001001	1035	429	Nuclear Regulatory Commission	Integrated Source Management Portfolio (ISMP)
☆	2492.	429-000001001	1035	429	Nuclear Regulatory Commission	Integrated Source Management Portfolio (ISMP)
☆	2493.	429-000001001	1035	429	Nuclear Regulatory Commission	Integrated Source Management Portfolio (ISMP)
☆	2494.	Total				

Lets start by staring this “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.

Apply a **text facet** to the *Unique Investment* column and select all the rows that have the value “Total” and **star** these. 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 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.

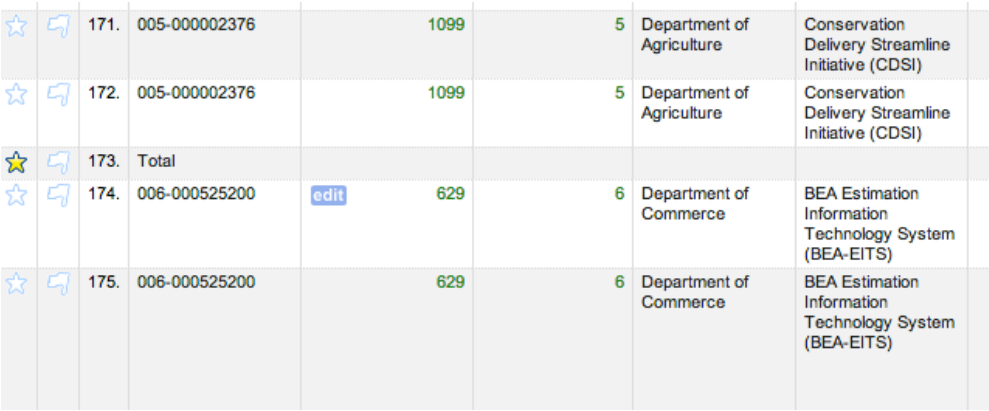


Unique Investment Identifier change invert reset

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



☆	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)

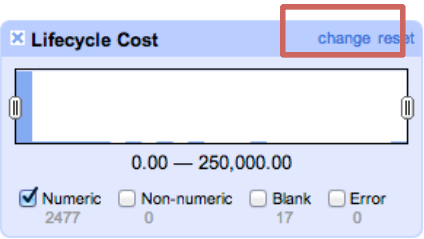
From the data displayed it looks like the totals are per agency. This can be confirmed by looking at how many agencies there are using another couple of facets. When happy that the summations are understood, delete the total rows such that they don’t spoil the later processing.

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.

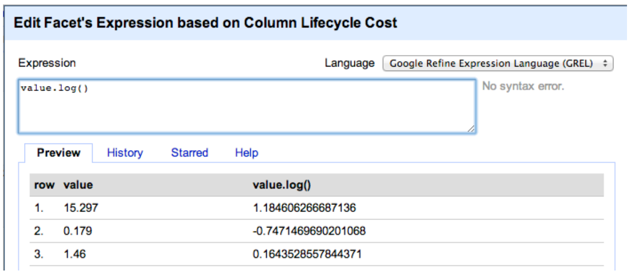
Applying a **numerical facet** to the *Lifecycle Cost* column is useful in some ways, but doesn't truly represent the distribution of values from a norm.

In order to distribute the values more evenly, click the change button. From the box that appears we can apply filters and programmatic changes to the values in the columns.



0.00 — 250,000.00

☑ Numeric 2477 □ Non-numeric 0 □ Blank 17 □ Error 0



Edit Facet's Expression based on Column Lifecycle Cost

Expression

Language: Google Refine Expression Language (GREL)

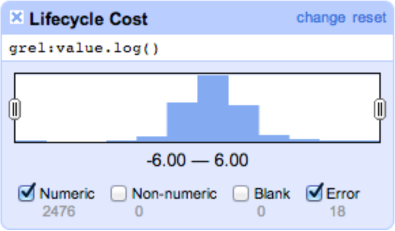
```
value.log()
```

No syntax error.

row	value	value.log()
1.	15.297	1.184606266687136
2.	0.179	-0.7471469690201068
3.	1.46	0.1643528557844371

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 look at the values of the outliers to discover if there are errors in the dataset.

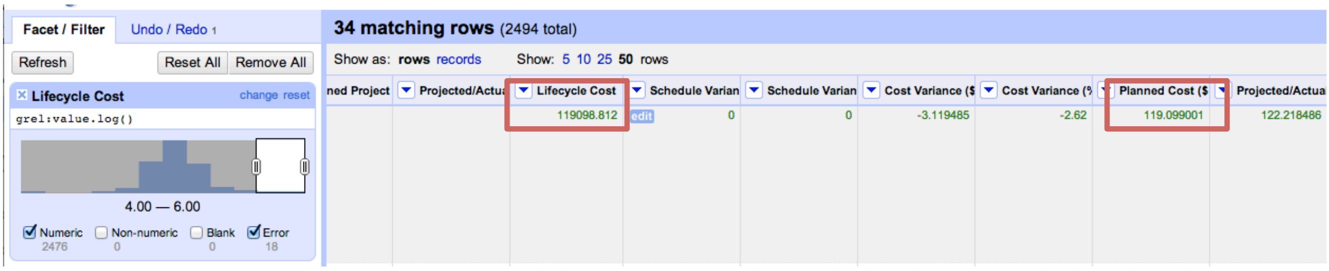


grel:value.log()

-6.00 — 6.00

☑ Numeric 2476 □ Non-numeric 0 □ Blank 0 ☑ Error 18

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.



grel:value.log()

4.00 — 6.00

☑ Numeric 2476 □ Non-numeric 0 □ Blank 0 ☑ Error 18

34 matching rows (2494 total)

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

ned Project	Projected/Actua	Lifecycle Cost	Schedule Varian	Schedule Varian	Cost Variance ($	Cost Variance (%	Planned Cost ($	Projected/Actua
		119098.812	0	0	-3.119485	-2.62	119.099001	122.218486

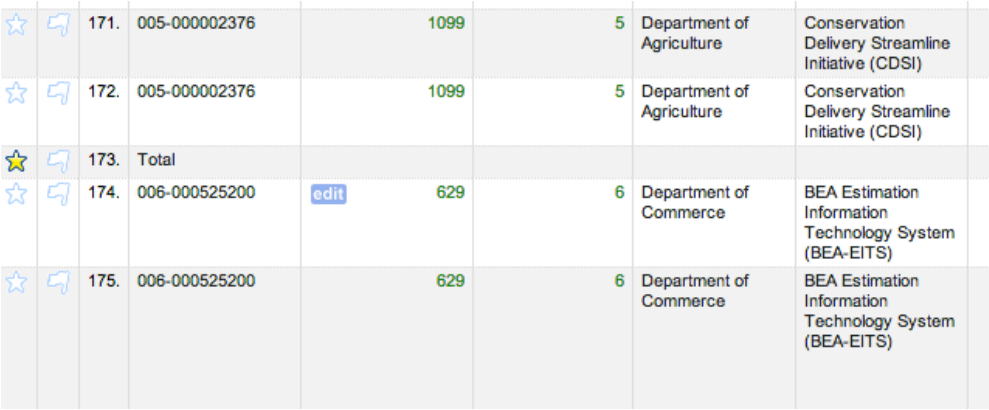
Looking at the different between lifecycle cost and planned cost should reveal the extent of the problem and allow it to be fixed.

Imagine the knock on effect this had with the totals!

N.B. While the totals rows were added for the purposes of this exercise. This record is the original!

Redundant Data (Dataset 2)

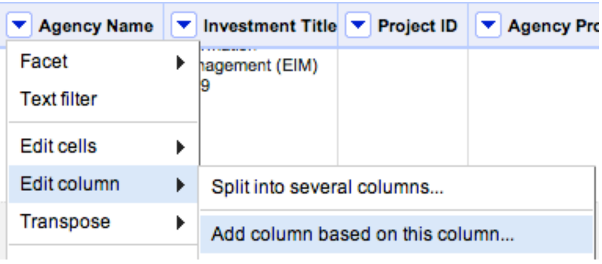
During the summation records exercise, it was discovered that the data appears to be grouped by Agency.



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	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)

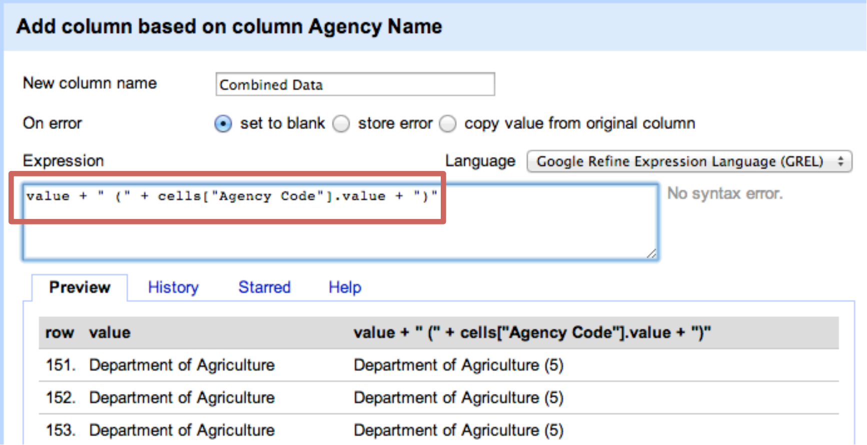
Looking at this data again, it should also be clear to see that we have an *Agency Code* and *Agency Name* columns. While it shouldn't matter that we have both pieces of data, redundant data can also lead to errors. Beneficially, redundant data can often be easier to fix; the more data you have, the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



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

Language: Google Refine Expression Language (GREL)

```
value + " (" + cells["Agency Code"].value + ")"
```

No syntax error.

Preview | History | Starred | Help

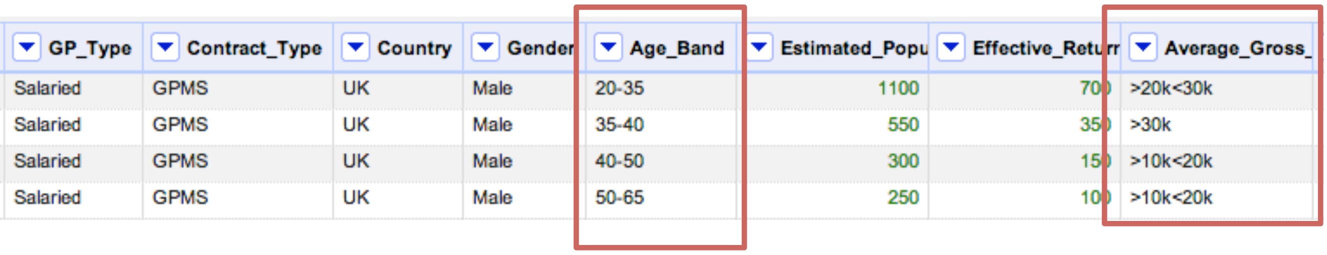
row	value	value + " (" + cells["Agency Code"].value + ")"
151.	Department of Agriculture	Department of Agriculture (5)
152.	Department of Agriculture	Department of Agriculture (5)
153.	Department of Agriculture	Department of Agriculture (5)

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.

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).



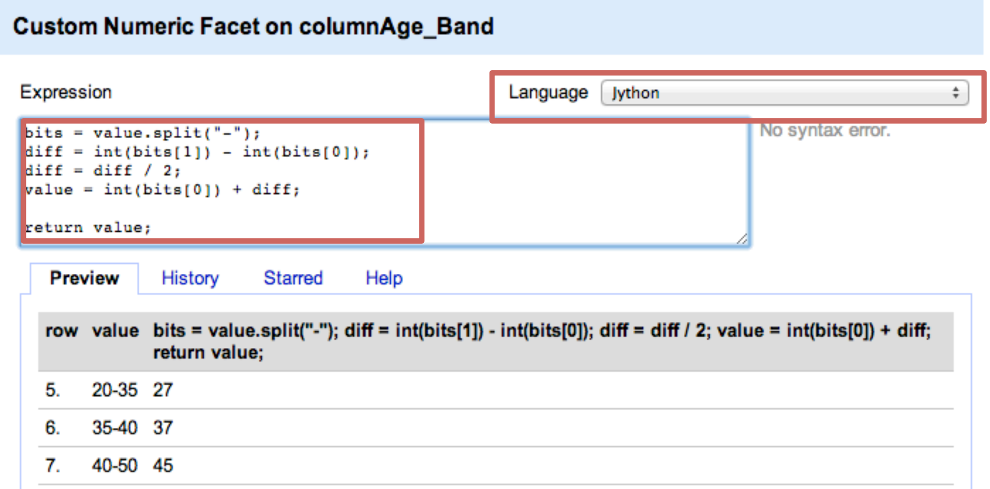
GP_Type	Contract_Type	Country	Gender	Age_Band	Estimated_Popu	Effective_Returr	Average_Gross_
Salaried	GPMS	UK	Male	20-35	1100	700	>20k<30k
Salaried	GPMS	UK	Male	35-40	550	350	>30k
Salaried	GPMS	UK	Male	40-50	300	150	>10k<20k
Salaried	GPMS	UK	Male	50-65	250	100	>10k<20k

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 it 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 **jython** 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

Language: Jython

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

return value;
```

No syntax error.

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);
```