A FOLLOW-UP GUIDE TO OPENREFINE

"A TOOL GUIDE IS A WORLDVIEW" EDITION by Eliza B
JUNE 2020

A NOTE ON CHOOSING SOFTWARE FOR YOUR PURPOSES

Someone asked on the discussion board about whether it's possible to use Excel for data clustering in preparation for data cleaning. The answer is yes, in the same way that it's possible to use a screwdriver for hammering a nail into a wall: clumsily, inefficiently, and you might damage the screwdriver, the nail, or the wall. By the same token, OpenRefine can create pivot tables of a sort, but it's cumbersome and confusing and much smoother to do them in a spreadsheet program (e.g. Excel, OpenOffice Calc, or Google Sheets). Meanwhile, there will be other functions that multiple types of tools do equally well (Both OpenRefine and various spreadsheet software make it very easy to split columns based some character, for example.)

My philosophy on tools is not to be too attached to any of them for their own sake, but to use the best tool for the job, and keep your data in a nimble, open form (i.e. a CSV) that that permits easy transfer between them. Use the tools and don't let the tools use you.

A NOTE ON OPENREFINE IN PARTICULAR

It's always good to understand a bit about how your software started life, and who owns and maintains it now. OpenRefine was born as GoogleRefine, one of the many projects that Google has developed, released to the wild, and then dropped because it wasn't profitable. (Remember GoogleReader? Google+?) But the uproar among data users when they pulled back Refine was enormous because Refine was so incredibly useful, and there was nothing else like it to be had for love or money. So Google finally relented and handed over its code to become opensource software.

Opensource software consists of code that is available to anyone to inspect, to replicate, and make new versions of. When Google abandoned and opened its Refine sourcecode, the software began a new life with a new name and is now maintained by volunteers. (This is why some older documentation for OpenRefine has the Google logo on it, and why the language for writing transformations still bears the name Google Refine Expression Language (GREL).)

OpenRefine is both *free* and *opensource*, but note that the two terms are not synonymous. When Refine belonged to Google, it was free, but not opensource. When the code that software is built with is held secret by its owner, we call it *proprietary*.

SOME NAVIGATION BASICS

Filtering. A useful tool that we didn't look at in class is the Filter. Click the drop-down arrow next to any column name that contains textual data, and choose `Text filter`. A filter box appears on the left. Type any string in the box, and all the rows that do not contain that string in that column will disappear.

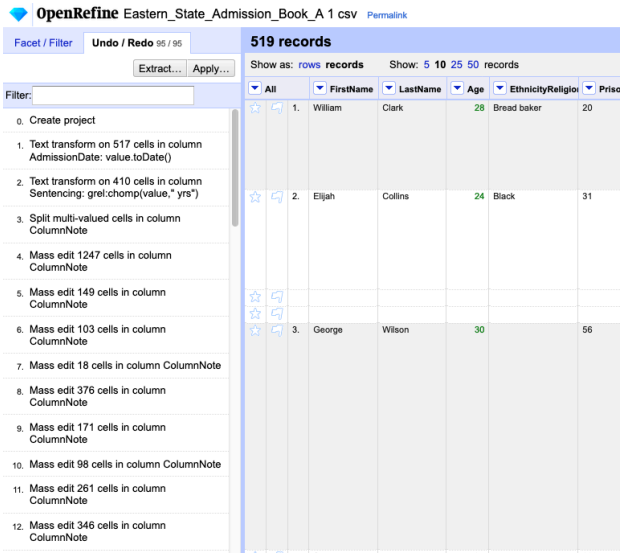
Filtering is great for exploration. For example, try filtering for "soul" in the [`Description`] column. You can quickly take a look at all the entries where the record-keeper referenced a prisoner's soul.

Now X out the Filter box (upper left), and start over with a clean left-hand panel. Create a Text facet from the [`EthnicityReligionOccupation`] column. Then again create a Text filter for soul from the [`Description`] column. *Note how the entries and numbers in the Text facet box change!* The facet now applies only to the entries that you have filtered out.

Filtering is also useful for editing. Any edits you make to records while your data is filtered applies *only to those records.* After you X out the filter, the edits you make to the filtered records remain while the other records reappear and are unchanged.

Undo/Redo. After any action you take, you go back in time to undo the action. In fact, you can undo as many immediately previous actions as you want.

Over on the left-hand panel, Click the `Undo/Redo` tab. There you'll see a list of all the actions you've taken so far. Click on any of them to roll back that action and all subsequent actions.



Export Your Data. Once you have your data in a form where you want to export it and use it elsewhere (to make a visualization, for just one example), click the button in the

upper-right that says `Export`. You'll get a drop-down with various file formats. You'll probably want to choose `comma-separated value` (csv), but you have other options as well.

Note that the `Export project` choice is for a time when you might want to export your whole OpenRefine project (including a history of actions to Undo/Redo) to share with someone else or open elsewhere. If you just want the data, and not the whole OpenRefine project, choose one of the options in the second section of the drop-down.

Steps Toward Cleaning the Eastern State Penitentiary Data

This is not an exhaustive list of how to clean the data. Rather, it's a few specific approaches I took to make the dataset that you practiced visualizing. As with many tools, there's more than one way accomplish most tasks. These are mine.

For additional friendly tutorials and guides, check out the resources I've posted on the W3 Github page. And remember that Google is literally a great method for finding answers to your "how to" problems.

In the examples below, [`words in square brackets`] refer to COLUMN names.

Prepare [Sentencing] for numerical/quantitative analysis. Lengths of sentences are listed mostly as "X yrs", where X=a number. To remove the " yrs" so that we can compare these values mathematically, use the following steps:

- From the drop-down menu next to the name [`Sentencing`], choose:
  `Edit cells > Transform`

- In the "Expression" box, write this bit of code in GREL:
  `value.chomp(" yrs")`

- This code says: "The new value is equal to the existing value, with the characters (aka substring) " yrs" removed, ***IF*** the *final* four characters of the original string are in fact " yrs". Otherwise, no change to the existing value. Also, both the original value and the new value are strings."

  You can see a different definition of `chomp` here.

- Click `OK`.

- From the drop-down menu next to the name [`Sentencing`], choose:
- `Edit cells > Common transformations > To number`

Create new column for "intent."

- From the drop-down menu next to the name [`Offense`], choose:
  `Edit column > Add column based on this column`

- In the new dialogue box, give your new column a name in the `New column name` bar. (Something like: `Intention`)

- In the "Expression" box, write this bit of code in GREL:
  `if (value.contains("intent"), "intention ascribed", "no intention")`

  This code says: "The new value in each cell (a string) will be one of two possible options. If the original value (also a string) contains the substring "intent" *anywhere* within it, then the new value will be "intention ascribed." If "intent" does not appear anywhere within the original value, then the new value will be "no intention."

  Note that this bit of code combines *two* different GREL functions. Read about the `contains` function. Read about the `if` function.

- Click `OK`.

Create new column for property/violence.
This one is a little more complicated. We'll go in a couple of steps.

- The first step is to look at the data and see what patterns exist and how we can break down the problem into smaller steps. I created a text facet on the [Offense] column, and ordered them by count. Skimming through the list, I saw that just four words (murder, manslaughter, rape, assault) could catch all the violent crimes. Catching all the property crimes would take a lot more words.

  (Note that I did this *after* cleaning up typos, capitalizations, spelling, and the like, as we did in class!)

  Here's how I proceeded:

- From the drop-down menu next to the name [`Offense`], choose:
  `Edit column > Add column based on this column`

- In the new dialogue box, give your new column a name in the `New column name` bar. (Something like `ViolenceTrue`)

- In the "Expression" box, write this bit of code in GREL:

```
or ((value.contains("Murder"),
(value.contains("Manslaughter"), (value.contains("Rape")),
(value.constains("Assault")))
```

Note that once again, this code combines two different functions: the `contains` function (which we saw above) and the `or` function.

The `or` function is a Boolean function, rather than a string function, as we've seen so far, or a mathematical function. A Boolean function doesn't return a string or a number, but either the value `true` or the value `false`.

This code says: If the original value (which is a string) contains any one of these four substrings, then the new value will be `true`. If none of these substrings are present in the original string, then the new value will be `false`.

Note that when you're dealing with strings, capitalization matters! So if you specify "Horse stealing" as a string that you are looking for, then your code will not match "horse stealing."

- Click `OK`.

- Now we have a column full of trues and falses, but we need labels that actual specify "property" and "violence" for the column to be useful for our human brains.

  From the drop-down menu next to the name [`ViolenceTrue`], choose:
  `Edit column > Add column based on this column`

- In the new dialogue box, give your new column a name in the `New column name` ban. (Something like `Property/Violent`)

- In the "Expression" box, write this bit of code in GREL:
  `if((value==true), "violence", "property")`

  The `if` function should look familiar! This code says that if the original value is `true`, then the new value should be `"violence."` If the original value is `false` (or anything else), then the new value should be `"property."`

  The difference between *this* `if` statement and our previous one is that the statement we want the computer to evaluate this time is a Boolean instead of a string. There are no quotation marks around `true`, because in this case, `true` is not a string; it's a Boolean value. If we wanted to evaluate a mathematical statement, it might look like this:
  `if((value==54), "this is 54", "this is not 54")`
  or

`if((value==54), 54.00, "this is not 54")`
Numbers, like Boolean values, do not get enclosed in quotes.

- Click `OK.`

- Now you can delete the first column you created if you want to. From the drop-down menu next to the name [`ViolenceTrue`], choose:
`Edit column > Remove this column`

- IMPORTANT TO NOTE! There is probably a way to achieve this task quicker. There is probably a way to do it by creating just one new column rather than two. But it would have taken me longer to figure out how to do that than to just do it in the slightly longer way that I did. As you learn any new code or technical skill, this will often be the case. There will be multiple paths to your goal, and it really doesn't matter if you get there in the absolute most efficient way, as long as you're learning and eventually getting there.

Standardize Geographic Names. When creating a strategy to standardize geographic names, I again created a Text Facet on the [SentencingLocation] column, and took stock of what's there. I saw that the vast majority of locations were either "`Philadelphia`" *or* the name of a Pennsylvania county (e.g. "`Centre`", `Chester`", "`Bucks`") followed by "`Co.`"

I know that in order to geocode (that is, convert to latitudes and longitudes) a list of locations, we need to create a standardized format for those locations. For locations in the U.S., the names of cities should be followed by a comma and the name of the state. County names should have the full word "County" spelled out after its name, followed by a comma and the name of the state.

- To add "`, Pennsylvania`" to every instance of "`Philadelphia`", it's easiest to edit the value right in the Facet box. Create a Text Facet for [SentencingLocation]. Click on `Philadelphia` in the list of values. A small blue `edit` button appears to the right. Click it, and fill in your new value (`Philadelphia, Pennsylvania`) in the box that appears.

- Here's my GREL for fixing the County entries:
`value.replace(value, "Co.", "County, Pennsylvania")`

  Read more about the `replace function`.

Standardizing the [Column Note] Column. This one is kind of hairy, and I only got started on it – I didn't go all the way. But demonstrating what I did to get started might be helpful.

The first thing to notice is that this column includes multiple types of information all smooshed together in different orders. Some of the information is related to literacy (*e.g. reads and writes, can't read, reads a little, spells, learned to read in prison,* etc.) some related to family and marital status (*single, married, not married, mother living,* etc.), some related to drinking (*drinks a little, sober, doesn't drink, drunkard, drinks wine, drinks occasionally*, *drinks sometimes,* etc.). These little bits of information are, for the most part, separated by commas.

- First, let's say we want to create a controlled vocabulary for each of these very uncontrolled categories. If I want to create two categories, literate and illiterate, for example, the first step is to separate out these different *segments* of information.

  Luckily for us, the data is entered in a way that makes separating out segments of information fairly easy.

- Click the down-arrow next to the column name.
  `Choose Edit cells > Split multi-value cells…`

- In the dialog box that appears, enter a comma: `,`
  Click `OK`.

  Note that you can enter any string in that box to indicate where the values should be split.
  Alternatively, you can enter a regular expression, and check the little box that tells OpenRefine that that's what you're doing.

- Now something very important has happened: you can examine your data as either **Records** or **Rows.**
  Please read this description of records and rows from Library Carpentry. It explains the differences pretty well.

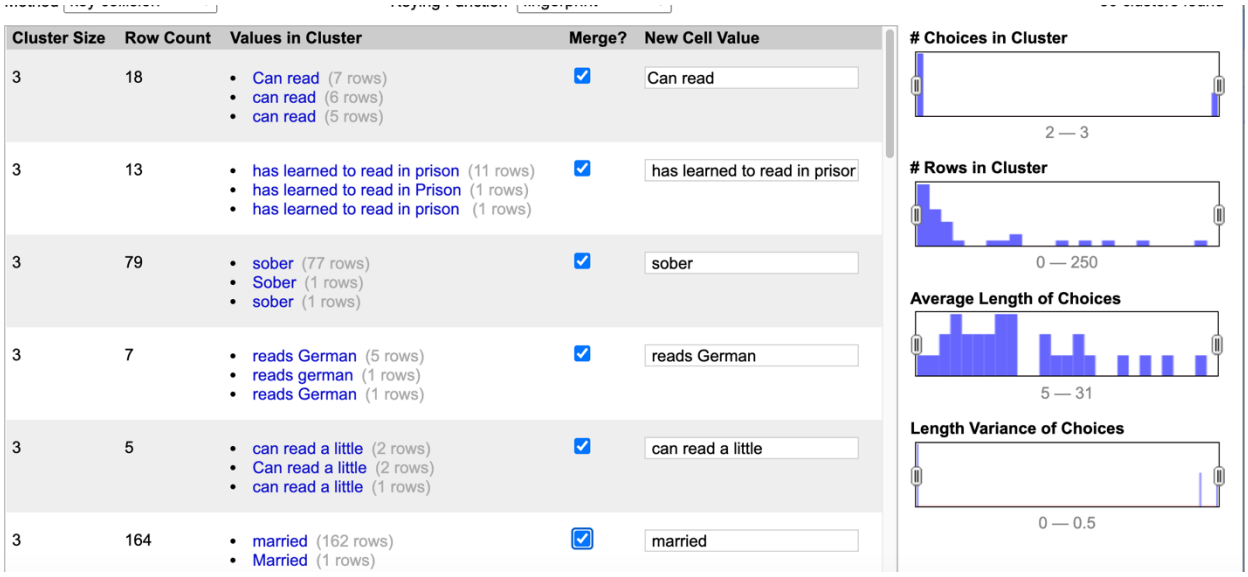
- Now we can deal with those little individual pieces of information separately. Run a `Text facet` on [`Column Note`], then `Cluster`.

- Now we can standardize, as in previous cases, in whatever way we choose. I decided to make very broad categories (literate; illiterate) (drinks; sober) to collapse many different fine distinctions.

- After my categorizing was finished, I put everything back together with:
  `Edit cells > Join multi-value cells…`
  You can use any separator you want to join everything back together (e.g. comma, semi-colon), as long as that character doesn't appear elsewhere in your data in that column.

- For the dataset I gave to you, I then split the rejoined column into separate columns, using:
  `Edit column > Split into several columns…`
  I was hoping that this would result in three columns showing, roughly: literacy, drinking habits, and marital/family status.

- But the data just wasn't that neat, and those three columns don't work too well. If I wanted to perform analysis on this data, and still wanted to produce three separate columns with proper data in each, I would try this:
  - Undo the column split I just did.
  - Create new columns on the original column using some GREL code and the .contains function.
  - As an exercise, try this if you want! How does it work? Do you have any other ideas for how to clean this column of data?



Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
3	18	• Can read (7 rows) • can read (6 rows) • can read (5 rows)	☑	Can read
3	13	• has learned to read in prison (11 rows) • has learned to read in Prison (1 rows) • has learned to read in prison (1 rows)	☑	has learned to read in prisor
3	79	• sober (77 rows) • Sober (1 rows) • sober (1 rows)	☑	sober
3	7	• reads German (5 rows) • reads german (1 rows) • reads German (1 rows)	☑	reads German
3	5	• can read a little (2 rows) • Can read a little (2 rows) • can read a little (1 rows)	☑	can read a little
3	164	• married (162 rows) • Married (1 rows)	☑	married