Keith Knop, UGA Libraries
ORCID: 0000-0001-7394-1897

URL for this handout: http://goo.gl/ZYeKSH

Introduction to OpenRefine

Sample Data Set

This demonstration uses a list of titles published in the pulp magazine *Weird Tales,* which originally ran from 1923 to 1954. The data was taken from Wikisource: https://en.wikisource.org/wiki/Weird_Tales

Download as Excel spreadsheet: http://goo.gl/XWVeLN
Download as tab-separated text: http://goo.gl/DQgsUG

OpenRefine can sometimes be fussy about Excel for no obvious reason; if you have difficulty getting it to recognize Excel data, try saving the file as CSV or tab-separated text.

This demonstration covers:

- Using "Split into several columns" to ensure each column contains one type of data
- Using "Split multi-valued cells" to keep multiple instances of the same kind of data in the same column
- Using facets to explore data and limit our actions to a subset of records
- Using clustering to identify and resolve inconsistencies
- Using GREL for basic data manipulation
- Augmenting data with information from external sources

Undo/Redo

OpenRefine's undo feature is incredibly robust, which is good if you, say, accidentally blank out half your data and don't realize it for a while. The Undo/Redo tab at the top of the left-hand panel contains a list of every operation you've performed since you created the project, and you can roll back to that step by clicking on it. If you undo and then do something new, the new action will replace all the subsequent steps.

Also note the **Extract...** and **Apply...** buttons. Extract will output a list of all (or some, if you choose) of your edits in a JSON format, which you can copy/paste and save in a text file. This is handy if you need to do the same processes on multiple datasets; you can simply paste your saved edit history into the Apply window to repeat the same actions on a new project.

GREL

OpenRefine provides several tools for manipulating data, one of which is the General Refine Expression Language, or GREL. The syntax is very similar to working with formulas in Excel. Some commonly used GREL functions (such as trimming whitespace or converting to upper, lower, or title case) have shortcuts under **Edit cells > Common transforms**.

GREL assumes that anything outside of single or double quotes is either a variable or a function. There are several predefined variables, but the most used one is **`value`**, which simply refers to the current

value of each cell in a column. You will probably also often want to refer to a value in a different column, which you can do using **`cells["Column Name"].value`**

Here's an example GREL function from the OpenRefine documentation, which takes two arguments (the parts in parentheses):

> **chomp(string s, string sep)** Returns a copy of `s` with `sep` removed from the end if `s` ends with `sep`; otherwise, just returns `s`. For example, `chomp("hardly", "ly")` and `chomp("hard", "ly")` both return the string `hard`.

A few points to note:

1. You will probably not often need or want to pass in two literal values; you're much more likely to use one or more variables. For example, **`chomp(value, ".")`** will remove the ending period from any cell in the current column that ends with a period.

2. Unlike Excel formulas, for nearly any function, you can pass in the first argument from outside the parentheses by linking it to the function with a period: **`value.chomp(".")`** has the same effect as **`chomp(value, ".")`**. Useful because...

3. You can link multiple functions together, so that the result of the first becomes the input of the second. For example, **`value.trim().chomp(".")`** will first remove any leading or trailing whitespace from the current cell contents, and then remove any final periods. This lets you do several things in one step and also helps reduce the infinite rat's nest of nested parentheses you can end up with in Excel formulas.

Regular Expressions

Several GREL functions for manipulating strings support regular expressions (or regex), a syntax for matching *patterns* rather than literal character strings. They are incredibly powerful and potentially incredibly frustrating and (fortunately for me) are way out of scope for this presentation. However, here's an example of a function that uses regex to convert a pattern like Firstname Middlename Lastname to Lastname, Firstname Middlename.

```
value.replace(/(.*)\s(.*)/, "$2" + ", " + "$1")
```

In GREL, regular expressions must be surrounded by forward slashes. A period in regex is a wildcard for any character, and * indicates zero or more of the preceding character, so **`.*`** is a pattern that will match anything; **`\s`** matches a space. Because the default behavior of **`.*`** is "greedy," the first will match as much as it possibly can, up to the final space in the cell. The inner parentheses are capture groups: matching material inside the first and second set of parentheses is remembered, and then recalled later using **`$1`** and **`$2`**.

Note this isn't perfect! It doesn't know about "Jr.," for one thing, so there will still be some manual cleanup required--but at least all the juniors will be faceted together.

Keith Knop, UGA Libraries
ORCID: 0000-0001-7394-1897

URL for this handout: http://goo.gl/ZYeKSH

For more on regular expressions, I recommend https://regexr.com/, which includes documentation and allows you to test and troubleshoot expressions on the fly.

Resources

OpenRefine Documentation and Introductory Tutorials

Wiki
https://github.com/OpenRefine/OpenRefine/wiki
OpenRefine Recipes
https://github.com/OpenRefine/OpenRefine/wiki/Recipes

Reconciliation Services

VIAF, ORCID, and Open Library
http://refine.codefork.com/

Geonames
https://github.com/cmh2166/geonames-reconcile
http://christinaharlow.com/walkthrough-of-geonames-recon-service

LC Name Authority File and LCSH
https://github.com/cmh2166/lc-reconcile

FAST Headings
https://github.com/lawlesst/fast-reconcile

Various Lessons

Library Carpentry
http://data-lessons.github.io/library-openrefine/

OpenRefine for Ecology
http://www.datacarpentry.org/OpenRefine-ecology-lesson/

Cleaning Data with OpenRefine
https://programminghistorian.org/lessons/cleaning-data-with-openrefine

Fetching and Parsing Data from the Web
https://programminghistorian.org/lessons/fetch-and-parse-data-with-openrefine

Using OpenRefine with MARCEdit
http://blog.reeset.net/archives/1873

Regular Expressions

https://github.com/OpenRefine/OpenRefine/wiki/Understanding-Regular-Expressions
https://regexr.com/

Presented for the University of Georgia Libraries
Professional Development and Research Committee