**Learning Journal weeks 6-8**

**Ellen Kirkpatrick**

**Week 6**

Open Refine data carpentry lessons

Working with openrefine

1. start openrefine. double click on .exe. file. Got taken to a blank black screen with lots of output getting written. It took some time but then openrefine came up in internet explorer.
2. Clicked create project, get data from, this computer. Selected SAFI messy file (had downloaded from data carpentry website). Uploaded data. Clicked create project.
3. Use faceting to look for potential errors. Scroll to village column, click down arrow and chose facet, text facet. Left panel a box appeared with all unique values in village column.
4. Sorted by name and count. data is problematic as is not consistent and some have spelling errors. Some have text names, others have number names.
5. Hovered mouse over names in facet list. Edit function becomes available.
6. Edited the errors so entries are all consistent. The name 49 cannot be made consistent with other data.
7. Applied text facet to interview date column. There are 19 options available. Column is formatted as text.
8. Produce timeline display for interview\_date. Select edit cells, common transfers, to date. Column converted to dates. Most dates were collected in November 2016.
9. Clustering - in village facet in left panel, click cluster. In pop up, selected key collision method and metaphone3 keying function. 2 clusters identified.
10. Click merge box beside each cluster and click merge selected and recluster.
11. Changed spelling of names, more clusters identified.
12. **Note:** clicked on the merge more than once, solutions for later exercises may be different.
13. Transforming data. Click items owned column, edit cells, transform.... New window opens. Can type general refine expression language (GREL).
14. Remove all left square brackets. In expression box type value.replace(”[”, “”), click ok. No longer left square brackets in item\_owned column.
15. Use same strategy to remove single quote marks, right square brackets in items\_owned column. Used the following GREL as directed in solution: value.replace("'", ""), value.replace("]", ""), value.replace(" ", ""). All on separate lines. Mostly successful, but can still see occasional right bracket next to a value. Otherwise, all replaced with semi-colons.
16. Ran value.replace("]", "") for a second time on its own. Successful. All right brackets disappeared.
17. Use text facet to see what items were commonly owned. Click items\_owned, choose facet, custom text facet. Type value.split(”;”) in expression box. Click ok. Box appears on left panel with all items owned.
18. Which 2 items are most commonly owned? Sorted by count. Mobile phone and cow plough.
19. Which 2 are least commonly owned? Solar torch and solar panel.
20. Used same steps, cleaned the months\_lack\_food column. Got error at last custom text facet.**Error-** Left panel appeared saying: *Parsing error at offset 12: Missing number, string, identifier, regex, or parenthesized expression.* How to fix???? Not sure where I went wrong or how to recover steps.
21. Copied and pasted the commands from solution earlier to get exact typing right for the months\_no\_water column. Successful.
22. Added GREL language. Text facet appeared on left.
23. Repeated cleaning and GREL steps for liv\_owned, res\_change, no\_food\_mtiigation columns. All successful.
24. Repeated GREL command using history tab. Clicked reuse next to command.
25. Using undo and redo. Click undo/redo on the left. All operations/commands are listed.
26. Clicked on a number of past commands to see the change in the dataset. Tried to see where I want wrong with the error in months\_lack\_food column but could not work it out.
27. Trim leading and trailing whitespace. Create new text facet for respondent\_wall\_type. Edited data so that spelling was correct and condensed identical choices.
28. Remove whitespace. Choose edit cells, common transforms, trim leading and trailing whitespace. Only 4 choices in text facet in left panel.

Filtering and sorting with openrefine

1. Click on arrow next to respondent\_roof\_type, select text filter. Facet appears in left.
2. Type mabat, press enter. The column shows the rows with matching info. There are 58.
3. Change view to show 50 rows. Will see most of matching rows.
4. What roof types are selected? Mabatisloping and mabatipitched.
5. How would you restrict this to only one type? Be more specific with choices of letters/word in filter.
6. Excluding entries. Create text facet for respondent\_roof\_type. Drop down menu facet, text facet. Facet appears in left panel with 2 entries that agree with text filter.
7. Use include/exlude to select entries from one of these roof types. Hovered mouse over the entry and clicked include on mabatisloping. Other entry is excluded.
8. Sort. Select drop down arrow in gps\_altitude column. Select sort by numbers and smallest first. Column sorted in order. First few values are 0. This may be problematic, maybe missing data?
9. Clicked sort again, and selected reverse to reverse sort. Successful.
10. Sorting by multiple columns. Sort on gps\_longitude with number largest first. Successful.
11. Sort gps\_lattitude as number with largest first. Successful.
12. Select drop down arrow on village column. Select edit column, move column to end. Column moves to end so it can be compared easier with GPS coordinates.
13. Find village 49. Can’t see what village it is based on coordinates.
14. Sort interview\_date column by data. Select sort, date.
15. Move village column to start. Drop down, edit column, move to beginning.
16. Row for 49 corresponds with interview times for Chirodzo village. Unsure about GPS coordinates.
17. Open text facet for village. Drop down arrow, facet, text facet. In left panel, select edit on 49. Change 49 to Chirodzo to condense entries.

**Proof of Concept - ongoing, week 6 & 7**

Made a new project on github. But at first it was in ellenkirkpatrick, not the FOAR705 team. Had to create a new repositry within FOAR705 and then make a project within that. Not sure how really to use the project management tool at this stage.

Making notes in a word document at the moment for possible user stories. Not sure how many are needed, unsure how categories work.

Will transfer these notes onto overleaf at a later stage once ideas are more developed. Once on overleaf will copy over any additions to github for version control.

—>

Started proof of concept on overleaf. Structure: listed user stories under category headings and acceptance critera. Committed to github. Created a new repositry and within that, started a new project. Deleted previous repositry so that everything (overleaf commits and project) in the same repositry.

Added in user stories to github project. Did not add acceptance criteria, not sure if meant to.

User stories are:

**Identify commonalities:** As a research student, I would like a program which can compare multiple sources at once to identify common themes and terms. This will help determine relevance of source to project.

**Extract Information:** As a research student, I would like a storage program that can extract source and metadata from an online platform so I can create references and citations more efficiently.

**Store sources:** As a research student, I would like a program that can store multiple sources with metadata so I can access all research material in the same place.

**Annotations:** As a research student, I would like to be able to add annotations to stored sources so they can be referred to more easily in the future.

**Grouping:** As a research student, I would like to link specific sources through tags or labels so they can be grouped together.

**Export metadata:** As a research student, I would like to export metadata to a word processing program, such as Microsoft Word, in order to automate the reference list.

Acceptance criteria:

Identify commonalities:

* Find peer-reviewed journal article sources on multiple databases.
* Download sources to computer.
* Upload sources to Voyant.
* Use the trends and cirrus tools on Voyant to identify key themes and terms.
* Use reader tool to read abstracts and determine relevance.

Extract information:

* Open Zotero application on computer.
* Use contexts tool on Voyant to connect to Zotero.
* Extract source and metadata from Voyant onto Zotero for storage.

Store sources:

* Check that the extracted source appears in Zotero.
* Click on each source to ensure metadata is available.
* Click on each source to make sure the original website or pdf can be accessed.
* Flag source if metadata or part of metadata is missing, or if the source is not available.

Annotations:

* Add annotations to sources in Zotero.
* Access annotations in future, and add to them more.

Grouping:

* Add tags or labels to source according to relevance or specific theme.
* Search for specific groups of sources by these tags.

Export metadata:

* Export source metadata from Zotero to Microsoft Word.
* Check that metadata is correct.
* Add to bibliography list if required.

Pre-requisites:

* All user stories must be completed in order for full efficiency.
* Voyant must have been tested prior with known sources to determine whether this is reliable. (completed in elaboration)
* Zotero must be installed on computer and a connecting icon added to the browser of choice (Mozilla Firefox) before beginning. (completed in elaboration)

Quality assurance

I don’t understand this component of PoC, or how to link it to the user stories. Is this a process like elaboration - testing the functionality??

Started to use the project management on github, but can’t see how to do this within the project tab. It seems like just columns where you can move tasks. Might just list out tests in learning journal?

Did add an initial issue identified in elaboration, that urls cannot be copied to Voyant. It works better with uploading pdf sources. I added this in the issue tab in PoC repositry, and will continue to add issues. But this tab is not linked with the project tab.

Proof of concept - update 2

Got some feedback from Osmond after I offered a few comments on his proof of concept design. He recommended to specify what user stories were essential for the PoC and which were just extra. —> took on feedback and created a new section “Notes on User Stories” in overleaf draft.

Essential user stories - identifying commonalities and storing sources. These user stories link to the central aim of being able to compare multiple sources at once and storing them in a single place for more efficient use.

Extra user stories - extracting information and exporting metadata are also crucial to the PoC in terms of functionality. But they are there to make the entire process more efficient. The annotations and grouping are not essential overall but help make the process more impressive as they are helpful for researchers referring to work in future.

After completing this section, committed to github.

Quality Assurance

Still not sure exactly what I need to do in this section but specified what the tool/program needs to do to be successful. They are the following:

Identify commonalities - For Voyant to be successful, it must identify commonalities between the different sources. At least 3 key terms will be identified and will be reflected in the trends tool. It must also show where in the source is each term used.

Extract Information - For Zotero to be successful, it must be able to extract both the source and the metadata from Voyant in one process. There must not be two separate processes for extracting the source and the metadata.

Store Sources - For Zotero to be successful, the source and metadata should be stored within the program library. They should be able to be accessed at future points and the original source (whether it is a website, or a file) can be accessed.

Annotation - For Zotero to be successful, annotations should be able to be added to specific sources. These annotations should be saved and can be edited, or updated at future points of time.

Grouping - For Zotero to be successful, tags should be able to be added to sources depending on relevance, topic area or the needs of the researcher. A search of these tags in Zotero should produce all sources under this tag.

Majority focus on zotero, but as the identifying commonalities user story is identified as crucial, it means Voyant is required to be successful.

Commited to github after writing this section.

Created a second project in github, just for quality assurance tests. Don’t have time to complete these tests before Friday. But can enter them in a to-do list. Made blank project. The user story project is in Kanban, and I am having difficulties modifying these settings and don’t know how to use cards. So thought blank may be more applicable. Put all tests into to-do column.

Still can’t see how to export a report, but I downloaded a tex. file from the repositry and added to my proof of concept submission folder on cloudstor.

Committed updated journal to github.