Welcome to The Carpentries Etherpad!

This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

Use of this service is restricted to members of The Carpentries community; this is not for general purpose use (for that, try https://etherpad.wikimedia.org).

Users are expected to follow our code of conduct: https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

---

Library CarpentryNetwork of the National Library of Medicine
Online
Nov 12-13, 2020
9:00 am - 4:30 pm CST
Instructors: Sarah Lin, Benson Muite, Daniel Wheeler, Thomas Guignard
Helpers: Kim Pham, Clara Turp, John Orazem

Workshop Website: https://labs.timtom.ch/2020-11-12-NNLM-online/

If you have not completed pre-survey, please do it now!

Pre-survey: https://carpentries.typeform.com/to/wi32rS?slug=2020-11-12-NNLM-online
Post-survey: https://carpentries.typeform.com/to/UgVdRQ?slug=2020-11-12-NNLM-online

Zoom Link:

Day 1

Please sign in
Name / pronouns / location / operating system

- Dorothy/she,her,hers/Alabama/macOS Catalina V 10.15.7
- Laura Simpson/she,her,hers/Alabama/Windows 10
- Kiara Comfort/ she, her, hers/Iowa/Windows 10
- Will Dean/ he,him,his/Philadelphia, PA/ Windows 10
- Amy Nadell/she.ella/Phoenix, AZ/Windows 10
- Emily Kilcer/she, her/Williamstown, MA/macOs High Sierra 10.13.6
- Heather Moberly/she, her, hers/College Station, TX/Windows 10
- Jason Curtis/he,him,his/St. George, UT/Windows 10
- Caleb / he, him, his / Toronto, ON, Canada / MacOS Catalina
- Amy Suiter/she,her,hers/St. Louis, MO/Windows 10

- Amanda Haberstroh/she, her, hers/East Carolina University (Greenville, NC)/Windows 10
- Beth Ten Have, she, her, hers/Children's Hospital of Philadelphia/Win10
- Brian Tober/he, him, his/Salt Lake City, UT/macOS Catalina
- Amy Hughes/she, her/Cincinnati/Windows 10
- Qianjin Zhang/she, her/Iowa City, IA/Windows 10
- Camri/she,her,hers/Salt Lake City/macOS Catalina
- Taneka Dixon she/her Decatur, GA Windows 10

The Unix Shell - Thursday AM

Sample data for this lesson: https://librarycarpentry.org/lc-shell/data/shell-lesson.zip

pwd = print working directory
cd = change directory - will take you to your home directory if you type it on its own
cd <folder name> = change directory to a particular directory

- if that directory name has a space, you have to put quotes around the directory name
- e.g. cd "shell lesson"

cd .. = change to the parent directory (one step up in the path)

ls = list the content of the current directory
ls -lh = same as ls with more readable output

To get help on a particular command:
man <command> (won't work on Windows)
or
<command> --helph

mkdir <directory name> = create a new directory

cat <file name> = output the content of a file to the screen
head <file name> = same as cat, but only displays the few first lines of a file

- head -n 8 <file name> = only display the first 8 lines of a file, etc.
- it also works on multiple files:
- head <file 1> <file 2> = displays the first few lines of <file 1> and <file 2>, etc.

tail <file name> = same as head, but only dispays the last few lines of a file

- tail -n 8 <file name> = only display the last 8 lines of a file

Hint: when typing the name of a file or a directory, type the first few characters, then hit the tab key - it will autocomplete

clear =clears the whole shell screen and presents a new line

* is a wildcard character

relative directory path: cd ../larry

"folder" and "directory" means the same thing!

mv = move/rename (this is a 'destructive' action, it can really cause you a problem if you're not paying close attention)

- e.g. mv <file1> <file2> will RENAME <file1> to <file2>
- BUT if <file2> is a directory, it will MOVE <file1> there
- e.g. mv <file1> <directory name> will MOVE <file1> inside <directory name>

cp = copy

- e.g. cp <file1> <file2> will make a second copy of <file1> named <file2>

To move your cursor around (might not all work on Git Bash/Windows):
ctrl-a = move the cursor to the beginning on the line
ctrl-e = move the cursor to the end of the line
ctl-b = move the cursor backwards (same as back arrow)
ctrl-f = move the cursor forward (same as forward arrow)

history = outputs a numbered list of recently used commands

- ! <command number> = re-run a particular command from that list

VARIABLE NAME> = some text

- Assigns some text to a variable name

$<VARIABLE NAME> : accesses the content of that variable

- e.g.
- NAME=Daniel
- echo "Hello, there $NAME"   - echo is a command to display something to the screen
    - output: Hello, there Daniel

up arrow: get previous commands in history

touch <file name> = creates an empty file called <file name>

rm = remove/delete (another command that requires some caution - file deleted using rm will be delted forever, not put in the trash! Some systems will not warn or prompt you to confirm before deleting)

When typing file paths

- . (the dot) is a shortcut for "the current position"
- .. (dot-dot) is a shortcut for the parent directory

How to loop in the shell - use a for block:

- v---- filename is a variable name - can be anything

```
for filename in *.doc
do
```

- echo "$filename"
- cp "$filename" backup_"$filename"

done

for filename in <list of files> = will loop through a list of files

- e.g. for filename in file1 file2 file3
- or, we can use a wildcard:
- for filename in *.doc
    - ^-- this will list all files that end in .doc

To display the first line of a bunch of files:

for filename in *.tsv
do

- echo "$filename"
- head -n 1 "$filename"

done

To display the first and last line of a bunch of files:

for filename in *.tsv
do

- echo "$filename"
- head -n 1 "$filename"
- tail -n 1 "$filename"

done

nano <filename> to edit a file (will create a new file if none exist of that name)

- This changes the command line to interactive mode - see list of commands at the bottom of the window
- Type text, use the down, arrow keys to move around - the mouse won't work
- Use ctrl-O to save (you'll have to confirm the file name)
- ctrl-X to exit

Lines that start with # are comments - will not be interpreted by the computer, this is for humans, you or others, to add helpful info about what the script is doing

nano my_bash_script.sh

This script loops through .tsv files, returns the file name, first line and last line of the file.

for filename in *.tsv

do

- echo "$filename"
- head -n 1 "$filename"
- tail -n 1 "$filename"

done

To run the script, type
bash my_bash_script.sh

Data wrangling
ls -lhS
The S here sorts according to file size

head -n 3 2014-01_JA.tsv
get first three lines in this text file
wc -l
wc *.tsv
word count of all files with extension .tsv

wc -l *.tsv > lengths.txt
put the output of word counts into a text file

cat lengths.txt
this gives the output, but not in a useful form

sort lengths.txt
Somewhat better formatting, but need to speficy how the
sorting should work

sort -n lengths.txt
Now specify sort by numerical value in the first column - this may
be operating system specific

sort -n lengths.txt > sorted-lengths.txt
put the output in a file

wc -l *.tsv | sort -n | head -n 1
avoid writing intermediate files by "piping" output

ls -l | wc -l
get number of files and directories in a directory,
first list files and directories, then count number of lines

```
date
# get the time and date

date > logfile.txt
# put this into the file logfile.txt overwriting any entries

date >> logfile.txt
# append the file, preserving any initial information

mkdir results
# create directory to store information

grep 1999 *.tsv
# find all lines that contain 1999 and print them out

grep -c 1999 *.tsv
# count number of times 1999 appears in each file with
# the extension .tsv

grep -i revolution *.tsv
# case insensitive search for every line that contains
# the sequence of characters revolution

grep -i revolution *.tsv > results/$(date "+%Y-%m-%d)_JAi-revolution.tsv
# put results in a file with the date as part of the name
# way of keeping track of when information was recorded

grep -iw revolution *.tsv > results/$(date "+%Y-%m-%d)_JAiw-revolution.tsv
# now ask for revolution as a word with space around it, not
# as part of a bigger word
```

Intro to Git -- Thursday afternoon

Git is version control software: locally through git bash and on the web with GitHub
Cheatsheet available at: https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet

Git is accessed through the Shell/Command line

```
mkdir hello-world
#create a new folder/directory to work in
```

git init = get started with git

```
touch index.md
#create a markdown file we can type in for the lesson
```

ls -a = shows all files, including hidden files (those with a . in front of the filename)

git status = shows the status of actions within git

nano index.md
edit the file and add some content

git add index.md
add the modified file to the Git staging area

git commit -m "Add index.md"
commit the changes to the repo

Using Github

Github is a website for hosting your Git repo and letting others access it

Create a new repo: click create new at top left
Add a repository name: "hello-world"
(Other options can be left as default)
Click "Create Repository"

(The following commands can be pasted from github after you create your repo)
git remote add origin https://github.com/my-user-name/my-repo-name.git
git branch -M main
git push -u origin main

git remote -v
show url of remote repo to which we are connected

nano index.md
make some more changes to the file

git diff
see the differences between the file in the repo and the new changes

git add index.md
git commit -m "Add some text to our file"
commit the new changes

git log
see a list of our commit history

git push
upload our changes to Github website

It's also possible to add a file on the Github website with the "Add file" button
Github support markdown formatting for the Readme.md file:
https://guides.github.com/features/mastering-markdown/

git pull
download latest version of the repo from the Github website

Brief descriptions of:
git push
git pull
git diff

Amy-Dorothy-Qianjian
git push push files from terminal onto internet
git pull from internet to terminal
git diff shows differences between 2 versions

Amy-Taneka-Caleb
git push: Pushing the changes from your staging area to the web repository
pull: Pull from the web repository to your local repository and into the working area
diff: difference between staging area and working area on your local machine

Breakout room 3: Jason, Camri, Amy, Kiara
git push: replicates the data from your computer to the website
git pull: the reverse - pulls the data from the website to your computer
git diff - shows the differences between the two

Breakout room 4: Emily, Brian, Beth
git push - Pushes the file on local machine/drive to remote GitHub server
git pull - Pulls to local machine/drive the file from remote GitHub server
git diff - Shows the difference between the file living in the GitHub repository and the file that you have pushed/pulled to your machine for changes before you commit it back to the repository

Github pages - https://my-user-name.github.io/hello-world
git checkout -b gh-pages
git push --set-upstream origin gh-pages

Forking a repo
Example repo: https://github.com/suitera/hello-world
Click on "Fork"
Make some edits to a file then create a Pull Request

RESOURCE ON GIT
https://happygitwithr.com

FEEDBACK
+ Having helpers was really nice, good ratio to learners
- Setup was a little confusing, had existing GitHub account, passwords
+ Liked that someone else took notes
- Lost command that generated lots of text, could not keep up, notes helped
+ Shell was helpful in making Git go faster
- More relevance to real world, examples of how can use in library
+ Learned new things about Git and the Shell

- A little fast, need time to absorb and be comfortable asking questions
+ Sufficient time built in, did not feel too rushed
- Can miss things when in a breakout room
+ Learning new tools, especially shell
- Wanted second monitor to be able to follow better
+ Etherpad was helpful for notes. Learned how to connect command line to GitHub/Git account

---

Day 2

Library CarpentryNetwork of the National Library of Medicine
Online
Nov 12-13, 2020
9:00 am - 4:30 pm CST
Instructors: Sarah Lin, Benson Muite, Daniel Wheeler, Thomas Guignard
Helpers: Kim Pham, Clara Turp, John Orazem

Code of Conduct: https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

Workshop Website: https://labs.timtom.ch/2020-11-12-NNLM-online/

Post-survey: https://carpentries.typeform.com/to/UgVdRQ?slug=2020-11-12-NNLM-online

---

Please sign in
Name / pronouns / location / Cats, dogs or both?

- Laura Simpson/she,her,hers/Birmingham,Alabama/cat!
- Heather Moberly/she, her, hers/College Station, TX/cats!
- Amanda Haberstroh/she, her, hers/East Carolina University Laupus Health Sciences Library (Greenville, NC)/both! :)
- Amy Hughes/she, her, hers/Cincinnati, OH/dogs
- Dorothy Ogdon/she, her, hers/ Birmingham, Alabama/cats
- Qianjin Zhang/she, her, hers/ Iowa City, IA/ cat!
- Emily Kilcer/she, her/Williamstown, MA/both
- Jason Curti / he.him.his / St. George, UT / dogs all the way!
- Beth Ten Have, Philadelphia PA, Cats!
- Amy Nadell / she, ella / Phoenix, AZ
- Kiara Comfort/she,her,hers/Iowa/ Cats
- Brian Tober/he, him, his/Salt Lake City, UT/dogs
- Caleb Nault / he, him, his / Toronto, Canada / cats ALL THE WAY
- Amy Suiter / she, her, hers/ St. Louis, MO/ dogs
- Will Dean / he, him, his/ Philadelphia, PA/ cats all the way

Introduction to Regular Expressions -- Friday morning

https://regex101.com/: website to test regular expressions with explanations.
https://regexper.com: builds a visualization of a regular expression that is really helpful for debugging
Regular Expressions Cookbook (2nd ed), by Goyvaerts & Levithan: organized by task or use case; check out from your local library (no need to buy)

Lesson: https://librarycarpentry.org/lc-data-intro/

Regular expression help searching for specific patterns (address, email, date).
There are different flavors/dialects of regular expressions depending on languages (i.e. Python, PHP)

Tokens/ metacharacters (characters that have a special meaning) : they tell the computer, "Hey, I'm looking for a particular pattern."

#Characters
[ho] = Matches either the lowercase letter h or o. vs ho would match the letters h and o.
[09] = matches either 0 or 9
[a-h] = any one lowercase character that is between a-h in the alphabet. This is a range
[0-9] = matches digits between 0 and 9.
[ah]{2} = any two lowercase characters between a - h in the alphabet
[a-zA-Z] = combines two ranges with all letters in both lowercase and uppercase.
[a-zA-Z0-9] = all letters and digits are matched.

#Other tokens:
* = zero or more times. (ab*c will match ac, abc, abbc)
+ = one or more times (ab+c will match abc, abbc)
? = between zero and one time (ab?c will match ac, abc)
{n} = the characters are present n times.
. = A period matches any character
\ = backlash allows to escape metacharacters. i.e. \. will search for a period.
No need to escape the period inside a range. i.e. [.] is an actual period. What is in a square bracket should be taken literally, there are some exceptions.

#Word character
\w = letters, digits and underscores. Equals to [a-zA-z0-9_]

#Spaces
\s = space, tab.
\S = everything except a space or tab.

#Digits
\d = any digits
\d{3} = 3 digits. Any digit that is repeated 3 times.
\D = everything that is not a digit

#Anchors: The beginning and the end of a string.
\b = word boundary.
    \bmark: Looking at the beginning of a word. Matches mark, market, marking
    \bmark\b: looking for mark at the beginning and end of a word = only matches mark
^ = start of a string (^example will match on "example number three", but not "This is an example")
$ = end of a string (example$ will match on "This is an example", but not "example number three")

What will the regular expression ^[Oo]rgani.e\b match?
- organize
- Organize
- organise
- Organise
- organihe

organization won't match. It matches all the way until the z. It needs to be between an i and a e, so it doesn't work.

#Lazy and Greedy quantifiers:
+ - greedy quantifier: trying to match as many times as possible.
+? - Making it lazy: matching as few characters as possible.

Exercices

1. What will the regular expression Fr[ea]nc[eh] match?
Breakout room 1: French, France, Franch, Frence
Breakout room 3: Matches French and France
Breakout room 4: French, Franch, France, Frence

2. How do you match the whole words colour and color (case insensitive)?
Breakout room 1: colou?r, set flag to Case Insensitive
Breakout room 3: col(o|ou)r
Breakout room 4: [Cc]olou?r

3. How would you match the date format dd-MM-yyyy?
i.e. match 13-11-2020 but not 1-11-2020 nor 13-1-20
Breakout room 1: [0-9][0-9]\-[0-9][0-9]\-[0-9][0-9][0-9][0-9]
Breakout 2: \d{2}-\d{2}-\d{4}
Breakout room 4: \d{2}-\d{2}-\d{4}
Breakout room 3: \d{2}-\d{2}-\d{4}, \b[0-3][0-9]-[0-1][0-2]-\d{4}\b - sort of identifies the range for days/months

4. How would you match publication formats such as
British Library : London, 2015
Manchester University Press: Manchester, 1999 ?
Breakout room 4: Not quite finished: [a-zA-Z]+:[a-zA-Z]+, ?\d{4}

or .* ?: .*, \d{4}

https://github.com/LibraryCarpentry/lc-data-intro/blob/gh-pages/data/swcCoC.md

Examples of RegEx in libraries: https://acrl.ala.org/techconnect/post/fear-no-longer-regular-expressions/

Matching email addresses:
- suggestions:
\w*@\w*. = it stops at the . (i.e. name@gmail.com, the com is excluded)
.*@.*\.* = Matches a paragraph containing an email.
\b.*@.*\.*\b = still matches a full paragraph (.* means any characters any number of times).
[a-z]*@[a-z]*[.][a-z]{3} = wouldn't match photo.studio. Need to capture more than 2 or 3 characters.
\w*@\w*\.[a-zA-Z0-9]{2,3}
\b\w*@\w*\.\w*\b = matches some emails, but not if there are periods in the username part of the email.
\b[\w.-_]*@[\w.-_]*\.\w*\b = this works
#Challenge: Specific enough to only match what you are looking for, but also general enough to include all edge cases.

Matching phone numbers:
A few hints:
- start without the area code
- then try including the area code - with and without parentheses
- try to add a country code (+1) as well

Optional: Country codes can have up to three digits! Can you make it work for international numbers too?

Breakout room 1: we got close but weren't quite able to get all of the numbers. This was the closest: \d{1}-\d{3}-\d{3,}-\d{4,}
Breakout room 3: we started to work on matching a number like +1 (530)341-3230, \+?[+\d]{0,2} \(\d{3}\).*\d{3}-\d{4}
Breakout room 4: \+?\d?[-\s]?[(]?\d{3}[-)\s]+\d{3}[-]\d{4}

You can use Regular Expressions in Google Spreadsheets using functions, for example REGEXEXTRACT

Resources

https://librarycarpentry.org/lc-data-intro/
https://regexcrossword.com/
https://www.pgdp.net/wiki/Regex_Cookbook
https://acrl.ala.org/techconnect/post/fear-no-longer-regular-expressions/
https://regex101.com
https://regexper.com/

OpenRefine -- Friday afternoon

Brainstorm: messy data problems

- compare one dataset to another
- standardizing phone numbers
- deduping
- missing data
- misspellings (s/z)
- call numbers
- view all the data
- colors/highlighting in spreadsheets
- bibliographic records
- standardized structure: column names, variation on field data, abbreviations, ISBNS, date formats
- title changes
- publishing changes
- UNICODE
- geospatial data
- symbols/non-standard characters
- Initial Assessing the entire library collection by subject headings including subscription and OA

Open Refine Notes

To download and install OpenRefine:
https://openrefine.org/download.html
Sample dataset: https://github.com/LibraryCarpentry/lc-open-refine/raw/gh-pages/data/doaj-article-sample.csv

OpenRefine accepts a variety of input formats:

- Comma-separated or tab-separated values (.csv or .tsv files)
- Excel
- JSON
- etc.

Typical tasks:

- Splitting values that are combined in one column, e.g. first name/last name
- Compare files

Even though OpenRefine runs in a web browser, it doesn't require an internet connection, it is running locally on your computer

Create project by clicking **choose files** and then loading
the file you downloaded doaj_article_sample.txt

Preview the file and ensure you choose csv option as a delimiter
At bottom, you can choose appropriate options for delimiters. Can
also choose other file types, json

Try to rename your files in a meaningful manner, possibly including
the date. Then create project - this ensures it saves as you
progress so that you have a history of your actions.

OpenRefine, has views for rows and records. A record
can have more than one row, for example can have
a row for all authors of a record of a journal article.

Split authors by using drop down menu
Edit cells -> split multi-valued cells
use the | symbol as a separator then click **ok**

Can now view from row perspective, many more
rows than columns.

Can also use join multivalued cells to join back. Check
number of rows to know which mode you are in. Use
| symbol as a separator since commas already used
to separate last name and first name
May need to use join multivalued cells multiple times
because may have several columns that need to be joined.

FACETING
Works at entire value of cell
Facet all ows you to look at a subset. Facet on publisher
Facet-> Text Facet, Can include, exclude, invert multiple values

NOTE: (blank) means values are missing or empty

FILTERING
Can work on any part of content of a cell
You can do a text filter if you click on a dropdown of a field. Possible option is to use regex or be case sensitive

NOTE: If you're faceting/filtering so you get ___ matching rows (____ total) --> you are only making changes on the subset (matching rows) of your data

OTher filters: Timeline, numeric, scatterplot

NOTE: Facets/filters are compounded, if you add more facets and filters your results will reflect all of the facets andfilters you've selected

NOTE: You can edit values in bulk by clicking on 'Edit', making your change and hitting 'apply'

Split multi-valued cells -> to get there use the dropdown of a field -> Edit cells and enter the separator that you want to use

CLUSTER AND EDIT
to get there use the dropdown of a field -> Edit cells, this option lets you find similar values in cells and merge based on similarity. You can change your cluster method, and function (more details here https://github.com/OpenRefine/OpenRefine/wiki/Clustering-In-Depth)

OTHER FUNCTIONS IN OPENREFINE
Move columns
Rename
Sort data -> sorts are temporary, unless you choose the option 'Reorder rows permanently'
You can undo/redo your actions, OpenRefine keeps the steps of what you applied to your data, you can also save your actions or copy them by using 'Extract'

- In your facet/filter You can set your choice count limit to show more values

TRANSFORM
Edit cells->Common transforms->Trim leading and trailing whitespace
Text transformations allow you to use GREL https://github.com/OpenRefine/OpenRefine/wiki/General-Refine-Expression-Languageto alter your data
Edit cells->Transform, you can replace 'value' which represents the value in your cell unchanged
e.g. value.toTitlecase() changes all of your values to a Titlecase format
.toTitlecase() is a GREL function https://github.com/OpenRefine/OpenRefine/wiki/GREL-Functions

SET DATE FORMAT
value.toDate("dd/MM/yyyy") tells OpenRefine is the format of the cell (a date field)
Edit column -> Add column based on this column to create a new column using another column's data with the option of changing the data, the original column is kept the same

CHANGE DATE FORMAT
value.toString("dd MMMM yyyy") changes your date to day month name year
GREL syntax for dates https://github.com/OpenRefine/OpenRefine/wiki/GREL-Date-Functions

FIND LAST NAME, FIRST NAME
Facet->Custom text facet...
value.contains(",").toString() is a boolean facet, you get true or false. it is looking for your values that contain a comma ","
use a "." to chain functions together, the second function toString changes the results to a string format

PUTTING YOUR VALUES INTO ARRAYS
an array is a list of values, refine represents arrays as ["value1", "value2", "value3"]
value.match(/(.*),(.*)/).reverse().join(" ") puts your values into an array based on regex putting the last name and first name into groups(.match(/(.*),(.*)/)), reverse the values in your array (.reverse()) and merges them back together (.join(" ")

sample data
10.14260/jemds/201513
10.3390/inorganics3040534
10.3390/hydrology2040289
10.3390/moleculres201219754
10.3989/egeol.42059.269

(10.\d*)\/(.*) -> using regex to split text into two groups

GREL MATCH
value.match(/.*(hyd).*/i) -> put your values into an array matching on hyd, use i flag after the last slash to make it case insensitive
value.match(/^(10\..*?)\/(.*)/)
in openrefine you have to finish your expression and match the rest of your value, even if you only want a portion of it
so to only get the first group
value.match(/^(10\..*?)\/.*/)

You can also use regex with text filter and facet

Resources

- Library Carpentry OpenRefine Lesson: https://librarycarpentry.org/lc-open-refine/
- Getting Started with OpenRefine: http://lgatto.github.io/OpenRefine-ecology/00-getting-started.html
- Cleaning Data with OpenRefine: https://programminghistorian.org/en/lessons/cleaning-data-with-openrefine
- Cleaning Data with OpenRefine: https://programminghistorian.org/en/lessons/cleaning-data-with-openrefine
- freeyourmetadata.org: https://freeyourmetadata.org/
- Data Munging Tools in Preparation for RDF: Catmandu and LODRefine: https://journal.code4lib.org/articles/11013
- OpenRefine blog posts from meanboyfriend.com: http://www.meanboyfriend.com/overdue_ideas/tag/openrefine/?orderby=date&order=ASC
- Identifying potential headings for Authority work using III Sierra, MS Excel and OpenRefine: https://epublications.marquette.edu/lib_fac/81/
- OpenRefine blog: https://openrefine.org/category/blog.html
- Clustering algorithm documentation: https://github.com/OpenRefine/OpenRefine/wiki/Clustering-In-Depth
- GREL documentation: https://github.com/OpenRefine/OpenRefine/wiki/GREL-Functions
- Crossref.org: https://www.crossref.org/

Wrapping Up

- One up (green), one down (red) per person: http://note.ly/tomtimtomch

- Post-workshop Survey: https://carpentries.typeform.com/to/UgVdRQ?slug=2020-11-12-NNLM-online
- Any trouble claiming CLE, please contact nto@utah.edu