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 etherpad.wikimedia.org).

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

Workshop website: https://scottcpeterson.github.io/2019-12-16-UCBerkeley/
Library Carpentry Lessons: https://librarycarpentry.org/lessons/

Sign In Here:
Zac Painter (Stanford, Instructor)
Scott Peterson (instructor) UC Berkeley
Rick Jaffe (UC Berkeley, Helper)
Ariel Deardorff (UCSF, Helper)
Celia Emmelhainz (UC Berkeley, Instructor)
Christina V. Fidler (UC Berkeley)
Becky Escamilla (UCB LIT)
Janine Gericke (Bancroft, janineg@berkeley.edu)
Kristina Bush (UC Berkeley Library)
Natalia Estrada (UCB Library)
Molly Rose (UCB Library)
Megan Van Noord (UC Davis)
Amelia Llorens (Stanford, llorensa@stanford.edu)
Kristen Greenland (UCB)
Marri Atienza (UC Berkeley Library)
Daina Dickman (Sacramento State, dickman@csus.edu)
Christy Navarro (UC Davis Blaisdell Medical Library, Cenavarro@ucdavis.edu)
Sam Teplitzky (UCB Library)
Stephen Kiyoi (UC Davis)
Yuko Okubo (UCB Library)
DeeAnn Tran (SJSU Library)
Ann Glusker (UCB Library)
Tuzun Guvener (Freelance Consulting, guvenerzt@gmail.com)
Connie Wong (Stanford, hongnei@stanford.edu)
Eric Fernandez (UCB LIbrary)
Tony Sucheston (UCB Library)

##########
Jargon Busting & Tidy Data

Jargon Busting (Terms to define):
-Git - version control software (for keeping track of different versions of your files/code ex: file_v1.txt is a basic form of version control)
-Repository - a folder for your stuff/code
-GitHub - a central online place to store/share code

- Unix: scripting language underlying your computer
- SHELL - a way of interacting directly with the computer - writing commands to tell your computer what to do
- terminal - the place where you type your commands to the computer

- - BASH - a kind of shell (bourne again shell)
- - Zsh - another kind of shell

- GUI: graphical user interface (clicking on folders/files with a mouse)
- pipes - a way to use the output of one set of commands as the input for another set of commands

- GREL - the programming language behind Open Refine (tool for cleaning data introduced on day 2)
- Regular expressions (regex) - shortcut commands to find and filter data - pattern matching
- Python - a programming language great for working with data science/big data/regular data
R - a programming language that started for statistics but is now very popular for a variety of uses - especially by librarians :)
- SQL - a programming language for working with databases

- Cloud computing - lots of computers networked together to give you more power
- Cluster computing - like the above, but more localized to one specific location

- open source - when you make your code available for free to others to use/build upon (open access code)
Libre--an ethos about contibuting and making sure it is always free.
Free--it is no cost, but has financial backers behind it. Can not develop your code.
Password manager: BitWarden
Regex--Regular expression--a search for a patterns
Rexex101: https://regex101.com/
grep--global regular experssions print

Tidy Data:

Download the data here:
https://librarycarpentry.org/lc-spreadsheets/data/training_attendance.xlsx

Save as (modify name) and note changes on copy. Do not directly edit the original sheet.
Notes tab - codebook- or readme-like...keep track of change there
Variables in columns
Observations in rows
Don't combine multiple pieces of information in the same cell
Leave raw data raw

PGR|PDRA|Other - mixed values in one cell - hard to keep track.
Cancelled should be its own column - color coding will get lost in export; can sort by value if its in a column; would be clear which table and row the value applies to
2016 tab: Split tables into two tabs so RDM training and Open access are on separate sheets

Combine tables across years
Combine workshops, too?

Add cancelled column, enter Y or N; clear formatting

(Don't forget to update notes tab!)

dates - five digit string to represent days since Jan 1, 1900. Excel has problems with dates prior to 1900

ISO-8601 - standard time and date format. Designed to reduce ambiguity when communicating across cultural barriers and machines. YYYY-MM-DD

Carpentries suggests splitting year, month and date into separate columns

Split values in RDM training column C - in Excel, Data> Text to table. Wizard appears; define delimiter = '|' (Add two empty columns to the right of column C to avoid overwriting data in columns D and E)

Other issues: undefined abbreviations in headings; combined values in Delivered by cell; use same number of decimal places in all numbers.

Empty values: blank, null. DO NOT use: 0, -999, 999, NA

##########
Unix (Shell)
folder to download for lesson: https://raw.githubusercontent.com/librarycarpentry/lc-shell/gh-pages/data/shell-lesson.zip

https://www.theverge.com/2019/6/4/18651872/apple-macos-catalina-zsh-bash-shell-replacement-features
(In Catalina, Apple replaced the Bash shell with zsh; older versions still use Bash)

REPL: Read, evaluate, print, loop (see, e.g., https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop)

MARCtools on GitHub - various command line (shell) utilities for MARC records (https://github.com/tomcrane/MarcTools ??)

/ Root Directory (the default, base place everything comes from)
~ Home Directory (where you usually go to first in a Terminal)
$ prompt to enter text
Commands:
pwd = print working directory - shows you where in your directory (folder and file) structure you are
ls = list contents of the current working directory

cd is a command to change directory. It takes "arguments" - i.e., additional information provided to the command. For example:
cd .. = moves you to the directory up one level in the folder tree
cd - = moves to previous directory. Use this to toggle between two directories
cd ~ = move to your home directory
cd = (i.e., without an argument) also a way to move to your home directory
ls -l = list contents in long list format
ls -lh = long list format, human readable
Arguments are separated by a space
use quotes around arguments that contain spaces, e.g. cd "evil plan to destroy the world"
For help: manual page (man command)
man ls (Mac)
ls --help (Windows)
>> On Mac, type q to quit out of man and return to prompt
clear = clears screen

Open up a GUI window showing the current directory:
open . (Mac)
explorer . (Windows)
xtg-open . (Linux)

*Using ls command**
Use the manual page or help pages and figure out how to list the files in a directory ordered by their filesize. Try it out in different directories.
Afterwards, find out how you can order a list of files based on their last modification date. Try ordering files in different directories.

ls -lSh. = list in long list form, Sorted by filesize, displaying human-readable filesizes
ls -lt = list in long list form, sorted by modification date

navigate to shell-lesson
cd ~/Desktop/shell-lesson

mk firstdir make directory, name it firstdir

tab autocomplete ... start typing, click tab, shell will autocomplete the argument up to the point that one or more files or folders share a common name, that is, moving from left to right, each character is the same). At that point, you can type another character or more to disambiguate the name, then hit tab again to autocomplete further.

cat = concatenate command
control - c = kills the current command

head [filename] = look at top 10 lines (by default) of a file. Empty lines count.
tail [filename] = look at last 10 lines (by default) of a file.

Hint: use tab autocomplete to save typing when entering file name

head -n [filename] = n argument determines number of lines to look at
tail -n [filename] works the same.

less [file] = read file.
within less:
G = move to end
g= move to beginning
q = quit less
for help: man less or less --help
less -N [filename] = number lines

Shell history
Use up and down arrows to move line by line through the commands that you have entered. Hit enter to replay a command

Wildcards: ?, * (shift+8)
? matches one character
* matches zero or more characters
Shell converts wildcard before passing argument to command. Command never sees the wildcard.

Rename a file: mv. (think of renaming as moving from an old filename to a new filename)
e.g., mv 829-0.txt gulliver.txt changes name of file 829-0.txt to gulliver.txt

Copy a file (cp)
Instead of moving a file, you might want to copy a file (make a duplicate), for instance to make a backup before modifying a file. Just like the mv command, the cp command takes two arguments: the old name and the new name. How would you make a copy of the file gulliver.txt called gulliver-backup.txt?

Renaming a directory
Renaming a directory works in the same way as renaming a file. Try using the mv command to rename the firstdir directory to backup.

Moving a file into a directory
If the last argument you give to the mv command is a directory, not a file, the file given in the first argument will be moved to that directory. Try using the mv command to move the file gulliver-backup.txt into the backup folder

To remove (i.e. delete) a directory or file: use the rm command.
NOTE: THERE IS NO UNDO!!!! Take care. Better yet, use the -i argument to be asked if you really want to delete. For example:
rm -i diary.html
remove diary.html? y (type y or n for yes or no)

history command shows the history of the shell session:
history
One history is displayed, scroll up to see past commands

Do a reverse-i-search through your history:

Control-R '[text-to-find].
Control-R to continue finding earlier matches.
Click return to replay the command that has been found.

Use ![line of command in history] to replay that command

To print out history to a text file:
history > history.txt

echo command
Prints what you type to the screen. Use quotes to contain text with spaces
Useful in a for loop for outputting current status of script to the screen

Variables
name=Scott creates the variable 'name' and gives it the value 'Scott'

$variablename refers to variable.
echo $name
echo "Finally, it is sunny on" $(date) (date is a built-in variable.)

loops - let us go through multiple files instead of just one

touch - a command to create a file

for loops:

- for filename in *.doc
- do
- echo "$filename"
- cp "$filename" backup_"$filename"
- done

the above copies a list of files that end in .doc and adds backup to the beginning

Complete the blanks in the for loop below to print the name, first line, and last line of each text file in the current directory.

```
for file in *.txt
do
    echo "$file"
    head -n 1 "$file"
    tail -n 1 "$file"
done
```

nano - a built in text editor
to save in nano: ctrl + o
to save in nano: ctrl + x

save our bash scripts as a .sh file

to run our program type bash and then name of the file: bash myfirstbashscript.sh
wow we wrote and ran our first script! :)

wc: word count command
wc -l *.tsv
now we are going to take the output and save that as a text file: wc -l *.tsv > lengths.txt

now lets sort that file by length of the line and then save that as a new file: sort -n lengths.txt > sorted-lengths.txt
now we want to do less work by using a pipe!

if we want to take the wordcount and then sort it we can do: wc -l *.tsv | sort -n
now we want to get the top line: wc -l *.tsv | sort -n | head -n 1
-
**Count, sort and print (fadedgtail -n 2

Counting number of files
Let’s make a different pipeline. You want to find out how many files and directories there are in the current directory. Try to see if you can pipe the output from ls into wc to find the answer, or something close to the answer.

ls | wc

Writing to files
The date command outputs the current date and time. Can you write the current date and time to a new file called logfile.txt? Then check the contents of the file.

date > logfile.txt

Appending to a file
While > writes to a file, >> appends something to a file.
Let’s append the current date and time to the file logfile.txt?

date >> logfile.txt

****Dates and OS X*****
OS X doesn't like spaces in filenames in command line so date has to be formatted
Example:
echo $(date +%Y_%m_%d)
write to file with current date example:
 grep i revolution *.tsv > "$(date +%y_%d_%m)_JAi-revolution.tsv"

grep!
- a useful tool for finding things in our files
to find where 1999 appears in our tsv files: grep 1999 *.tsv
to count the number of times 1999 appears in each tsv file: grep -c 1999 *.tsv
to find all instances of french or france in our tsv files: grep -iwEo 'fr[ae]nc[eh]' *.tsv

on macs: grep -E '\d{4}-\d{4}' 2014-01_JA.tsv >issns.tsv
on pcs: grep -P '\d{4}-\d{4}' 2014-01_JA.tsv >issns.tsv

grep -Eo '\d{4}-\d{4}' 2014-01_JA.tsv | sort | uniq | wc -l

time for a for loop!
```
for name in "Jo" "Meg" "Beth" "Amy"
do
echo "$name"
grep -wo "$name" littlewomen.txt | wc -l
done
```

Day two

Attendance
Scott Peterson UC Berkeley
Zac Painter (Stanford)
Eric Fernandez (UCB LIbraryIT
Kristina Bush (UCB)
Becky Escamilla
Daina Dickman (Sacramento State)
Amelia Llorens (Stanford)
Ann Glusker (UCB)
Kristen Greenland (UCB)
Marri Atienza (UC Berkeley)
Natalia Estrada (UC Berkeley)
Molly Rose (UCB)
Janine Gercke (UC Berkeley)
Tony Sucheston (UCB)
Connie Wong (Stanford)
DeeAnn Tran (SJSU)
Rick Jaffe (UC Berkeley, Helper)
Megan Van Noord (UC Davis)
Christy Navarro (UC Davis)
Stephen Kiyoi (UC Davis)
Celia Emmelhainz (UC Berkeley)
Anna Sackmann (UC Berkeley)
Yuko Okubo (UCB)

##########
Git
Good at tracking changes from multiple individuals, highlighting conflicts between individual contributions ("commits")
Create a folder (mkdir) on desktop (name it what you want), change (cd) to that directory

git init
git config --list - shows your global git configuration values

git config --global user.name "yourname"
git config --global user.email "yourgithubemail@domain.xyz"

##########
OpenRefine

Notes: https://librarycarpentry.org/lc-open-refine/
Download DOAJ file here: https://librarycarpentry.org/lc-open-refine/setup.html

http://http://127.0.0.1:3333/

Difference between rows and records. We will see this when we split Authors column in multiple rows:
Author dropdown > Edit cells > Split multivalued cells... [pop-up window appears]
by Separator ... enter '|' (without quotes)
Click ok

Note that new rows have been added, without a title... these rows belong to the record above

To join rows again: Author dropdown > Edit cells > Join multivalued cells ...
Enter Separator ... '|'
click ok

Facets - try these:
Publisher pulldown > Facet > Text facet
Unique values appear in Facet column at left
Select a facet to see records with that value
Click Edit to change this value in all records.
Type exclude to de-select that facet

License pulldown > Facet > Text facet

Language pulldown > Facet > Text facet
Select 'English'
Click 'Edit'
Change value to 'EN'
Click Apply to make a mass edit

Clustering:
Compare similar values within a column
To begin, let's resplit the values in the Author column
Author pulldown > Edit cells > Split multivalues cells ... by Separator '\'
To cluster: Author pulldown > Edit cells > Cluster & Edit

Q. Can I identify only the values that differ by a period after an initial and merge those?

To adjust columns:
All pulldown > Edit columns > Re-order / remove columns

To undo:
Click Undo/Redo (next to Facet/Filter)
Step up to step prior to one that you want to undo. All later steps will be undone, too.

On Undo/Redo screen, you can extract your steps to a file that can be replayed later.

Transformations:
e.g., splitting values in one column into multiples, standardizing date formatting, trimming white space, etc.
Uses GREL (General Refine Expression Language)

Publishers > Facet > Text facet
Note that MDPI AG appears twice, one with 3 entries
Select that facet.
Publishers > Edit cells > Common transformations > Trim leading and trailing whitespace

Also try:
...
Title > Edit cells > Common transformations > To uppercase
Title > Edit cells > Common transformations > To titlecase

Title > Edit cells > Transformations
Enter in text field:
value.toUppercase()
ok
value.toTitlecase()
ok

to see history:
Title > Edit cells > Transformations ... click History tab to see your previous transformations

Export button in upper right of screen to export the project or the data

Date transformations
Date pulldown > Edit cells > Transformations
GREL:
value.toDate("dd/MM/yyy")
ok

Working with data types other than Strings

To create a Date column that is more readable:
Date pulldown > Edit column > Add column based on this column...

Enter:
Column name: Formatted date
GREL:
value.toString("dd MMMM yyyy")
click ok

Boolean searches on Authors
Example:
Authors > Edit cells > Transformations

if(value.contains("Flavia"), "It's Flavia!", value)

To create a facet showing Authors' names with a comma:
Start by splitting Authors by pipe (see above)
Authors > Facet > Custom text facet...
value.contains(",").toString()

Click on True facet to see Authors' names that contain a comma

{Beware: The following notes get sketchy!}
Quick discussion of arrays [value 0, value 1, value 2 ...]
To see: Subjects > Edit cells > transform
value.split()

To split, sort alphabetically, then rejoin with a pipe
value.split("|").sort().join("|")

Changing last, first to first last
Start with facet showing Authors' names with a comma, select that facet (see above)
Authors dropdown > Edit cells > Transformer
value.match(/(.*),(.*)/).reverse().join(" ")
ok

{Notes are caught up again!}

Connecting to external data

From an external URL:
ex. Journal ISSN to get information from CrossRef

Select one row (click star at left of row)
All pulldown > Facet > Facet by star

Select that facet

ISSN pulldown > Edit column > Add column by fetching URL...

On that screen, give column name 'Journal details'; click 'show' link next to HTTP headers to be used ...; add mailto:[emailaddress] to User-agent line, e.g.
OpenRefine 3.2 [55c921b]; mailto:rjaffe@berkeley.edu

In 'Formulate the URLs to fetch' field, enter:
"https://api.crossref.org/journals/"+value
click ok

You should see a new column named "Journal details" with a mess of text in it.
To trim down the text:
Journal details pulldown > Edit column > New column based on column
Enter Column name "Journal title"
Enter GREL:
value.parseJson().message.title
click ok

Reconciling our data against an external source

We will use Viaf reconciliation server written by Jeff Chu (public service)
Reconcile publisher

Publisher pulldown > Reconcile... > Start reconciling
Add standard service
Enter URL: http://refine.codefork.com/reconcile/viaf

Select Viaf in screen that appears
Uncheck "Auto-match candidates with high confidence"
Click

Two facets appear. You can close 'Publisher: Best candidate score'

Publisher pulldown > Facet > Text facet
Select facet 'International Union of Crystallography
Click double check next to 'International Union of Crystallography' to accept our local name for all values

Click reset in facets panel to show all facets

To change all values at once:
Publisher pulldown > Reconcile... > Actions > Match each cell to its best candidate

Create column with VIAF ID:
Publisher pulldown > Edit column > Add column based on this column
Name the column VIAF ID
Enter GREL:
cell.recon.match.id

Rejoin rows into records with multivalued author fields (see above)

Now you can export cleaned data as a .csv file

You might want to export your steps at this point, too:
UNDO/REDO > Extract

Discussion of Open Refine extensions (google it)
Download and add to webapp folder where your Open Refine application files are located

Postworkshop survey link:
https://carpentries.typeform.com/to/UgVdRQ?slug=2019-12-16-UCBerkeley