**Learning Journal for FOAR705**

**Week 1:**

Objective: Restore a file from a 6 month old back up

Action: Search on Google Drive for “Knowledge of Self and Others” the Mind and World essay I wrote over a year ago

Error: None

Result: Success, document found and opened

Objective: Look at project management tools and form opinions

Action: Observe ‘project management’ I am currently doing for my wedding on shareable google sheets. Although the way it is formatted is useful for me to understand, it can be hard to communicate what I mean to other people. I briefly looked into Trello, Jira and Asana. The visual nature of Asana seems much clearer as a form of communication.

Error: None

Result: Asana definitely seems like a more useful project management tool than what I currently use.

**Week 2:**

**Overall Objective:** Figure out what LaTex is and how to use it.

Objective: Use LaTex

Action:

1. Create an overleaf Account
2. Create a “Test LaTex” file

Error: None

Result: Success

Objective: Play around with LaTex

Action:

1. Change “\author” to “Georgia Rutherford”.
2. Delete \maketitle to see what happens

Error: Need to hit “recompile” for the example document to update.

Result: Apparently deleting \maketitle deletes the text that was above it in the document. I am unsure why.

Objective: See what happens if I put text above the \begin document

Action: Type \section{Hello World} above \begin document

Error: Using “/” instead of “\” makes it plain text instead of a section.

Using “()” instead of “{}” does not work (not entirely sure why but it looked weird).

Result: It made a page with that section before the other pages. This included the title even though the text for the title was above the “\section{Hello World}. This must be because of the “\maketitle” command

Objective: Make paragraphs

Action:

1. Press enter twice to change line
2. Write “\paragraph{}” on a new line and then continue text on the line below to add a larger break.

Error: Need to do two breaks to change the line of the text (otherwise the text is written next to the other text).

Doing three breaks doesn’t change the line further (still just turns up underneath).

Writing “\paragraph” changes the text inside the “{}” into bold.

I don’t like the indentations on the document, not sure how to change

Result: Not entirely sure if above is the proper way to make paragraphs, but it seems to work. Need to research further.

**Overall Result:** I have a much better idea on how to use LaTex and the process has been much more simple than I anticipated. However, there is still a lot I’m confused about. For example, I am not extremely fond of the formatting and would like to learn more about how to change that. At this stage I probably know enough to be able to complete the first assignment.

**Overall Objective:** Do Scoping exercise in LaTex

Objective: Create a LaTex document on overleaf

Actions:

1. Open a blank text
2. Name it Scoping Exercise
3. Change “\author” from my email address to “Georgia Rutherford”
4. Recompile

Error: none

Result: Successfully made document

Objective: Create a list of headings and subheadings on my document

Actions:

1. Write “\section{}” for A Day in the Life, Pains, and Gains
2. Write “\subsection{}” for Pains I Encounter, Pain Relievers, Gains I Would Like, and Gain Creators
3. Recompile

Error: realised after that I can use the “rich text” to put headings and subheadings in even quicker.

Result: Headings and subheadings created

Objective: Enter relevant information under each heading

Action: Entered in text

Error: I realised partway through that I can use “Rich Text” to very easily add bullet points and some formatting.

Tried to put in quotation marks and one of them came out backwards. Need to use ‘ ’ and ` ` keys instead of “” in LaTeX

Result: Finished Exercise

**Overall Result:** Finished Scoping Exercise

**Overall Objective:** Complete Data Carpentry exercises

Objective: Identify what is wrong with this spreadsheet/the steps you would need to take to clean up the two tabs and put them together

Things that are wrong:

* Graph labeled Plots uses yes, y, no and N
* Uses a key that a computer won’t be able to translate (yellow in first tab, asterisk in second)
* Some of the cells are left blank
* Page 2 title is not in the graph
* Asterisk used twice in the second tab (could be confusing as to which they refer to)
* Second tab has a mix of numbers and words in graph (1, yes, no, yes)
* Spelling errors (errth) and different ways of spelling (Mabati\_sloping vs mabatisloping)
* Data errors (-99 rooms)
* Livestock owned and numbers should be separate

Steps needed to take to put them together:

* You would need to change so they are using all the same values (example: all yes/no or all numbers for livestock)
* You would need to correct spelling so there are no errors and everything matches
* Fix data errors
* Make changes on a separate document

Objective: make a list of some of the types of metadata that should be recorded about this dataset

* What counts as an item for items\_owned
* Who was interviewed and how they were chosen
* What does no\_meals mean (number of meals? Per day?)
* Wall type on what? (House?)
* What does no\_members mean? (number of members? of household?)
* Location of villages
* Which year were the months\_lack\_food in?

**Overall Results:** I have read to the end of formatting problems. I was surprised to see that what I thought was a data error, the use of numbers like -99, was actually meant to convey a null value. This highlighted to me the issue of being incredibly clear with your data. I was also interested to read about how merging cells can be problematic for the computer seeing associations between data. Out of all the problems listed, merging cells is probably the one I would be most likely to commit.

**Overall Objective:** Submit LaTex Scoping Exercise

Objective: Fix an error I just found in the LaTex code “overfull \hbox”

Action:

1. Google what “overfull \hbox” means and how to fix it
2. Add \linebreak to move the word “Throughout” so it isn’t split in two

Error: Initially tried to just move throughout to the line below but it caused an indentation. Using \linebreak moves the word down without causing the indentation.

Result: The error was just highlighting that the word “Throughout” was being split in two. This didn’t really bother me but it was cool to learn how I can add a line break without having the indentation. Having tested it further, I can add \linebreak twice and start a new paragraph without the indentation.

However, having thought about it a bit more I realise that \linebreak will add the line break there even if my formatting changes and there is no longer a need for the line break. I am unsure as to how to fix the issue of “Throughout” being split without causing this second issue.

At this stage I’m going to leave “Throughout” as it is and look more into how to fix this “overfull \hbox” issue for the future.

Objective: Download .tex file

Action:

1. Download PDF
2. Google how to download .tex file from overleaf
3. Download .tex file
4. Unzip file

Error: I was lost for a little bit about where to find the button to download the file as .tex since it wasn’t near the PDF download button. Turns out it was in the menu bar under ‘Source’

Result: Both PDF and .tex files downloaded

Objective: Submit Exercises

Action:

1. Rename from “main” to “GeorgiaRutherford-ScopingExercise”
2. Submit .tex on cloudstor
3. Submit PDF on ilearn

Result: Submitted

**Overall Result:** Submitted LaTex Scoping exercise

**Overall Objective**: Data Problems in philosophy

One problem with the use of data in my field is that sometimes data is ignored. It is common among philosophers writing on the topic of work to write about 'the end of work' in society as though it is inevitable. However, there are statistics that show employment is actually very stable at the moment. This issue is discussed in 'The Return of Work in Critical Theory' on page 36 by Christophe Dejours, Jean-Philippe Deranty, Emmanuel Renault and Nicholas Smith.

The second issue of data use in my field also comes from the topic of work. A large amount of data exists around people's experiences of work and some of this data is contradictory or can be interpreted in different ways. This issue is discussed by Marie Jahoda in 'Work, Employment, and Unemployment'.

**Overall Result**: success

**Week 3**

**Overall Objective:** Complete Data Carpentry activities for this week

Objective: Extract the components of the date to new columns

Action:

1. Title columns Day, Month, Year
2. In B2 write Day=($A2)
3. Drag from the corner of B2 to the bottom of the column
4. In C2 write Month=($A2)
5. Drag from the corner of C2 to the bottom of the column
6. In D2 write Day=($A2)
7. Drag from the corner of D2 to the bottom of the column

Result: Success. I was previously unaware of the Day=, Month=, and Year= functions.

Objective: add another data point in the interview\_date column by typing 17/11

Action:

1. Write 17/11
2. Continue the day/month/year columns to include the new entry

Result: Success, A16 now displays 17/11/2019. I was also previously unaware excel auto-completed dates like this.

Objective: Apply a new data validation rule to one of the other numeric columns

Action

1. Click on column G (rooms)
2. Click Data
3. Click Data Validation
4. Click allow - whole number
5. Restrict to between 1-30
6. Change input message. Title: Invalid number. Input message: Must be whole number between 1-30

Result: Success

Objective: Apply a new data validation rule to one of the other categorical columns

Action:

1. Choose column B: Village
2. Click Data
3. Click data validation
4. Click allow - list
5. In source type God, Rauca, Chirodzo
6. Change input message. Title: Invalid Village. Input message: Only God, Rauca and Chirodzo accepted

Objective: Export data

Action:

1. Click File
2. Click save as
3. Change format to .csv

Error: For the SAFI\_dates file an error popped up saying I couldn’t save both tabs. This was okay because I only used one of the tabs. Still good to note for future reference.

Error: when I reopen the document it doesn’t appear to have saved the data validation I did, so I will try again.

Result: I think You need to save the document as an .xlsx file to save data validation, and then as .csv so you have access to the data if excel stops supporting the .xlsx files

Objective: Export the csv. View it in a text editor like Atom.io, Sublime Text, or notepad++ Think about the benefits of an always-readable and not tied to a subscription or specific program data format.

Action: I don’t have those programs downloaded on my computer, but I opened the .csv in just normal notepad and it was still recognisable. The .xlsx file, on the other hand, was complete gibberish. Having a file not ties to any specific program would mean that you can access it if you lose access to the program used, and that it is shareable with everyone no matter what program they use.

**Overall Result:** Data Carpentry Task Done

**Overall Objective:** Complete Scoping II: Computational Analysis in LaTex

Objective: Create document

Action:

1. Open Overleaf
2. Click new project
3. Click blank project
4. Title: Scoping II: Computational Analysis

Error: None

Result: objective complete

Objective: Create sections for Decomposition, Pattern Recognition, Algorithm Design

Action:

1. Write \section{...} for each topic
2. Click Recompile

Error: two of the topic headings did not appear as sections because without thinking I had capitalised the S is \Section

Solution: uncapitalise the S

Result: Success

Objective: Type information into each section

Action:

1. Add subsections for note organisation and referencing in each section
2. Write information

Error: None

Result: Success

**Overall result:** Scoping II complete

**Overall Objective:** Check out Bibtex

Objective: Create a bibtex file

Action:

1. Go to overleaf
2. Create a new project to test things in
3. Create a new .bib file

Error: None

Result: Success

Objective: Cite something

Action:

1. Type in test.bib: @book{carruthers2011opacity,

title={The opacity of mind: An integrative theory of self-knowledge},

author={Carruthers, Peter},

year={2011},

publisher={OUP Oxford}

}

1. Type \cite{carruthers2011opacity} in main.tex file

Error: received error “You have cited something which is not included in your bibliography.”

Solution: I had to specify a .bib file and which bibliography style to use in the .tex file.

Action:

1. Type \bibliographystyle{plain}
2. Type \bibliography{test}

Result: Success. The bibliography appeared at the end of the document and the citation is in the main body of the text

Objective: Cite the page number as well

Action

1. Change \cite{carruthers2011opacity} to \cite[p.~31]{carruthers2011opacity}

Error: None

Result: Success. Citation now says [1, p. 31]

Objective: Change citation to a format more like my usual style.

Action:

1. Download an APA style bibtex style from <https://www.reed.edu/cis/help/LaTeX/bibtexstyles.html>
2. Upload file to project
3. Delete \usepackage[utf8]{inputenc}
4. Write \usepackage[natbib]
5. Change \bibliographystyle{plain} to \bibliographystyle{apa-good}

Error: Literally everything broke and it couldn’t compile. I clearly missed some steps. I’m going to take a break and come back tomorrow.

Objective: Start new bibtex test

Action

1. Open new project
2. Create .bib file
3. Put carruthers reference info into .bib file
4. Type \bibliographystyle{plain} in .tex file
5. Type \bibliography{test} in .tex file
6. Type \cite{carruthers2011opacity}

Error: Reference not found

Solution: Read my learning journal. Be confused because I did everything the same as last time. Realise that the .bib file is named Test.bib this time and that it is cap sensitive. Change to \bibliography{Test}

Result: Success

Objective: Change to APA style reference

Action:

1. Google how to use APA bibtex, click on <http://homepage.stat.uiowa.edu/~rlenth/ALPHA/apa-tutorial.pdf>
2. Write \usepackage[natbib]
3. Don’t delete \usepackage[utf8]{inputenc} this time
4. Change \bibliographystyle{plain} to \bibliographystyle{apalike}

Result: Success in changing the format of the reference. Although it came up as Carruthers (2011) and I would prefer (Carruthers, 2011).

**Overall Result:** There is definitely some promise here and I think I will find bibtex very useful in the future, but I am still interested in the digitising handwritten notes idea. So for now I am going to leave this and look further into character recognition.

**Overall Objective:** Check out existing written to digitised text software

Objective: Download something to test

Action:

1. Google ‘open source character recognition’
2. Find out about Google Tesseract from <https://opensource.com/life/15/9/open-source-extract-text-images> “The technology extracts text from images, scans of printed text, and even handwriting”
3. Click on <https://opensource.google.com/projects/tesseract>
4. Click on <https://github.com/tesseract-ocr/tesseract>
5. Download tesseract file
6. Unzip tesseract file

Error: Lots of files within the tesseract file. Will need to google how to actually use this software.

Objective: Download Tesseract and use it

Action:

1. Google how to install <https://www.bl.uk/britishlibrary/~/media/bl/global/early%20indian%20printed%20books/training%20resources/installing%20and%20using%20tesseract%20ocr.pdf>
2. Go to <https://github.com/tesseract-ocr/tesseract/wiki>
3. Then go to <https://github.com/UB-Mannheim/tesseract/wiki>
4. Download 64bit installer
5. Go to <https://github.com/tesseract-ocr/tesseract/releases>
6. Download zip of source code
7. Unzip Source code
8. Rename JPG file to test1
9. Put JPG file into tesseract-4.00.00alpha folder

Error: Can’t find the folder. Probably because I downloaded the newer version. I will try put the JPG file into the Tesseract-OCR folder instead

1. Open Command prompt

Error: At this stage I got very confused and asked my IT trained fiance with help on how to use command prompt. .

1. Firstly I need to be in administrator to have permission to edit files in program files folder. (Right click on command prompt icon, right click command prompt, click run as administrator)
2. Next I need to change directory from windows/system32 to the ‘working folder’ (Type cd "C:\Program Files\Tesseract-OCR")

Error: Can use tab to switch quickly instead of typing out everything

1. Next I need to type what I want it to run (tesseract.exe) what I want it to use as input (test1.jpg) and what I want it to call the output (outputtest). This looks like C:\Program Files\Tesseract-OCR>tesseract.exe test1.jpg outputtest

Error: At first I forgot what I called the jpg file and wrote test.jpg instead of test1.jpg so it failed and I had to go back and change it

Result: I successfully converted the text of the image into text. However the conversion itself was terrible. The phrase “Philosophical Paper -4500 words min” Was changed to “Vb) losoghicad Xager â€œ4500 words in”

Objective: Retry handwritten to digital with tesseract. This time by myself and with much more simple text

Action:

1. Rename JPG file to test2
2. Place test2.jpg in Tesseract-OCR folder
3. Open command prompt in administrator
4. Change directory to Tesseract-OCR
5. Type tesseract.exe test2.jpg outputtest2

Result: Successfully ran through the process by myself, but the test went even worse. The phrase “Hello World” was changed to an arrow pointing upwards.

**Overall Result:** After having done some more research it doesn’t seem like the technology is developed enough to be useful for me. I could spend a lot of time and effort to teach Tesseract my handwriting in particular, but even then I couldn’t find any evidence that it could be close enough to 100% effective at this stage. I think this task is just too large for me to complete this semester.

**Week 4**

**Overall Objective**: Scoping of my note taking process

After discussing my proof of concept with Shawn I have decided to look closer at my actual note taking process. I usually take handwritten notes while I am in class, for first quick readings of texts, or to organise my thoughts about essay structures. However, when I am doing a more close reading of a text I tend to move to a digital space as I am more likely to use this work in my essay drafts. For this analysis I am going to be focusing on note taking once it has reached this digital stage. Specifically I am going to analyse my note taking of Saul, J. (2012) Politically Significant Terms and Philosophy of Language

1. Open pdf
2. Open Google docs
3. Read through section
4. Copy interesting quote
5. Paste interesting quote into document
6. Note page number
7. In brackets note why quote is interesting
8. Make seperate section in document titles "Notes"
9. Under notes heading write my more general thoughts of the argument/how I would respond
10. Save document

**Overall Result:** This process wastes time switching between documents and copy/pasting. It is also sometimes difficult to see how my notes connect to the broader argument of the paper.

**Overall objective:** Check out hypothes.is

Objective: download hypothes.is

Action:

1. Go to site hypothes.is
2. Sign up
3. Download browser extension

Error: none

Result: downloaded

Objective: Make an annotation

Action:

1. Open PDF of Saul, J. (2012) Politically Significant Terms and Philosophy of Language

Error: Unable to annotate on page

Solution: First I have to click on the browser extension “h.” to turn on annotations for that page.

1. Select title
2. Click on the “ at the side of the page
3. In provided space write “title of the paper” as a test
4. Tag annotation with Saul, language, gender, sex

Note: Can’t annotate emails or google docs. Don’t think there is anything I can do to change that.

Note: The scroll at the side of the page shows exactly where they are and how many notations are above or below the scroll.

Result: Success, annotation made

Objective: Retrieve annotations

Action:

1. Go back to hypothes.is
2. Click on section of annotations just made

Note: Clicking on one of the tags takes me to a list of work other people have annotated using that tag

Result: It seems I can visit annotations in context (which takes me to the PDF I was reading) and share a link to the annotations with other people, but there doesn’t seem to be a way to download the annotations so I have access to them when I am offline. Downloading the PDF downloads it without the web browser add ons.

Objective: Export Annotations

Action:

1. Google Export annotations
2. Read <https://web.hypothes.is/blog/viewing-and-exporting-hypothesis-annotations/>
3. Go to <https://jonudell.info/h/facet/?max=50>
4. Copy “Share annotations” link
5. Past link into the URL section
6. Click HTML

Error: “Nothing found for this query”. This could have been for two reasons. One, I think I made my annotations private. Two, the PDF I was annotating required access to ILearn and this could have been blocking access.

1. Change annotations to public and try actions 4-6 again

Error: “Nothing found for this query”

1. Open a stanford encyclopedia of philosophy entry for self consciousness and make annotations
2. Repeat steps 4-6

Error: “Nothing found for this query”

1. Copy link of the stanford encyclopedia of philosophy web page rather than the “Share Annotations” link

Success: The site found my annotations

1. Click the save button

Error: Did not save

1. Turn off adblocker and click the save button

Result: Annotations Can be saved as either a HTML, CSV or JSON

Objective: Commit to GitHub

Action:

1. Rename file from “Hypothesis” to “Hypothesis-Test”
2. Open Github
3. Make “Notes” repository
4. Click “Upload Files”
5. Click “Choose your Files”
6. Choose file to upload
7. Create commit name and description

Error: Because it is a HTML file, it saves in GitHub as the web code rather than how I want to see it

Solution: According to: <https://stackoverflow.com/questions/8446218/how-to-see-an-html-page-on-github-as-a-normal-rendered-html-page-to-see-preview> , to view a HTML file I can paste the link to <http://htmlpreview.github.io/>

Error: While the above solution worked to a point, I was unable to expand some sections in the file through this viewer

Solution: I could redownload singular files to expand the sections and see what is in them. According to: <https://stackoverflow.com/questions/4604663/download-single-files-from-github?fbclid=IwAR2xbCoTISbhxiXH3QfkwE0CxzavIyGchUDapdKs18oPw7K9K_W2nT7iC0c>

To redownload singular documents within a repository I right click on “Raw” and click “Save Link As”

Result: HTML file successfully saved to GitHub and successfully viewed afterwards. However, this process is by no means quick and easy.

**Overall Result:** Hypothes.is seems like an extremely useful resource. I like the ease annotating on the document I am using, being able to see exactly where the annotations are in the document, clearly seeing how my annotations fit into the document as a whole, and being able to tag my annotations. Using Hypothes.is would save me a lot of time switching between documents to read and then take notes. It would also save me time copy/pasting quotes.

While the note taking process itself is much better while using hypothes.is, the saving process is more complicated. The process of exporting annotations is important to me because I want to be able to back them up and have access to them at all times.

Out of all my exporting options I find the HTML file to be the clearest to read, however this has issues when saving to GitHub. I currently rely on google drive for version control and back ups of my notes, however this also has issues with reading HTML files.

My note taking process using Hypothes.is would be:

1. Open document that needs annotating
2. Turn on hypothes.is browser extension
3. Highlight section of interesting text
4. Make a note as to why it is interesting in provided space
5. Tag annotations
6. Make more general notes in “Page Notes” section
7. Finish annotating document
8. Copy URL
9. Open <https://jonudell.info/h/facet/?max=50>
10. Paste URL in the space provided
11. Click Save document as HTML
12. Rename file from “hypothesis” to something more appropriate
13. Open GitHub
14. Open “Notes” repository
15. Click “Upload files”
16. Click “Choose your files”
17. Choose file to upload
18. Create commit name and description

To view the file:

1. Copy URL
2. Paste URL into <http://htmlpreview.github.io/>

Or redownload the file:

1. Right click Raw
2. Click Save Link As

Both of these options have issues. However, using hypothes.is to take notes could still be a good option if it were possible to automate steps 8-18 and have detailed commit names/descriptions.

For this task I would want a tool that:

1. Allows for user input of URL
2. Allows for user input of file name
3. Allows for user input of description
4. Opens <https://jonudell.info/h/facet/?max=50>
5. Enters URL
6. Creates HTML file
7. Saves HTML file
8. Renames HTML file to previously inputted name
9. Saves file to output folder
10. Opens github
11. Opens “Notes” repository
12. Goes to “Upload files”
13. Goes to “Choose your files”
14. Choose file to upload
15. Names previously input file name
16. Creates description previously input

By saving the file both on the computer and in github I would have ease of opening the file as well as having backups.

**Week 5**

**Note:** Having browsed the Facet website a bit I don’t think it has an API I can use for the above tasks. For this reason I am going to change my project to focus on getting my notes off hypothes.is and reformatting those notes to be human readable. For this task I would want a tool that:

1. Allows for input of url
2. Allows for input of file name
3. Interacts with hypothes.is API to get annotation information
4. Outputs annotation information in file
5. Reformats annotation information in a specific way

For future goals I may want a tool that analyses my notes and outputs trends. However, for now I will focus on the above process.

**Overall Objective:** Get some information off hypothes.is

Objective: Get API key

Result:I thought I needed an API key, but according to <https://h.readthedocs.io/en/latest/api/authorization/> “API requests which only read public data do not require authorization”. All of the annotations I have made so far with hypothes.is are public. Therefore I am going to try to continue without an API key. If it turns out I do need one there are two ways listed on this site on how to get them.

Objective: Use SWAN on Cloudstor

Action:

1. Open Cloudstor
2. Click on SWAN
3. Start session
4. Open Terminal
5. Use command ls and cd to move around

Result: I know how to use the terminal in SWAN. I will need to do more research in how to use R or Python on this, but I am happy for now that I know something, and am familiarising myself with the layout of SWAN.

Objective: Get some information off hypothes.is

Action:

1. Google hypothes.is API
2. This page <https://h.readthedocs.io/en/latest/api-reference/v1/#tag/annotations> has a section on getting annotations. It states the command GET /annotations/{id} and [https://api.hypothes.is/api/annotations/{id](https://api.hypothes.is/api/annotations/%7Bid)}. When I click the “visit annotation in context” link on hypothes.is, it changes the end of the URL to include /#annotations:K\_JrMsbiEem5hutSGynvDw. I assume this is the annotation ID
3. Type [https://api.hypothes.is/api/annotations/{K\_JrMsbiEem5hutSGynvDw](https://api.hypothes.is/api/annotations/K_JrMsbiEem5hutSGynvDw)} into a search bar

Error: Either the resource you requested doesn't exist, or you are not currently authorized to see it

Solution: Remove the {} and just type <https://api.hypothes.is/api/annotations/K_JrMsbiEem5hutSGynvDw>

Result: Success, I got some information off hypthes.is. It isn’t very clear, but I expected that. This is a section of it. {"id": "K\_JrMsbiEem5hutSGynvDw", "created": "2019-08-25T02:43:51.327033+00:00", "updated": "2019-08-30T08:46:31.519316+00:00", "user": "acct:georgiarutherford@hypothes.is", "uri": "https://plato.stanford.edu/entries/self-consciousness/", "text": "People to look at for my idea of the self and other: Fichte (1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a somewhat different perspective, Mead (1934; Aboulafia 1986)"

Objective: Get annotation information through the terminal

Action:

1. Open terminal in SWAN
2. Type get https://api.hypothes.is/api/annotations/K\_JrMsbiEem5hutSGynvDw

Error: bash: get: command not found

Solution: I must have made a sigh of defeat, because my fiance walked over had a look at what I was doing and said “try wget” which worked. Writing the “w” will make it get from the web. Solution is to write wget <https://api.hypothes.is/api/annotations/K_JrMsbiEem5hutSGynvDw>

Result: Success, the text output from the command was:

--2019-08-30 12:04:45-- https://api.hypothes.is/api/annotations/K\_JrMsbiEem5hutSGynvDw

Resolving api.hypothes.is (api.hypothes.is)... 104.20.215.15, 104.20.214.15, 2606:4700:10::6814:d70f, ...

Connecting to api.hypothes.is (api.hypothes.is)|104.20.215.15|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: unspecified [application/json]

Saving to: ‘K\_JrMsbiEem5hutSGynvDw.1’

[ <=> ] 1,915 --.-K/s in 0s

2019-08-30 12:04:46 (16.8 MB/s) - ‘K\_JrMsbiEem5hutSGynvDw.1’ saved [1915]

A file has been downloaded to SWAN, when I type ls I can see that it is there.

Note: I tried the exact same process on git-bash and it didn’t work

**Overall Result:** Very successful. Not only have I gotten information from hypothes.is, but I was able to do so using terminal.

**Overall Objective:** Reformat the file so it is more human readable.

Objective: Rename file K\_JrMsbiEem5hutSGynvDw

Action:

1. Type mv ~/K\_JrMsbiEem5hutSGynvDw ~/hypothesistestnotes.txt

Result: Success

Objective: Move hypothesistestnotes.txt into its own folder

Action:

1. Type mkdir Notes
2. Type mv ~/hypothesistestnotes.txt ~/Notes/hypothesistestnotes.txt
3. Type ls ~/Notes

Result: Success, Notes folder has been made and hypothesistestnotes.txt has been moved into it

Objective: Copy file

Action:

1. Type cp hypothesistestnotes.txt Notes

Error: I made a file called notes instead of making a file in notes. I renamed it to hypothesiscopy.txt

Result: Now I can mess around with it without changing my original file

Objective: Change file type to csv

Action:

1. Type mv hypothesiscopy.txt hypothesiscopy.csv
2. Open the csv document

Result: Successfully made a csv document. The way sections are separated because of commas is useful because the information is separated by commas, such as *"id": "K\_JrMsbiEem5hutSGynvDw", "created": "2019-08-25T02:43:51.327033+00:00"*. However, this is also problematic because the annotations and quotes themselves also use commas, such as *Hegel (1807; Pippin 2010), and, from a somewhat different perspective, Mead (1934; Aboulafia 1986)*.

**Overall result:** I changed the format so it was a little more human readable. It still isn’t very clear though.

**Overall objective:** make a script that will download annotations for me

Objective: Get first line of script to run

Action:

1. Type nano DownloadNotes.sh
2. Type wget [https://api.hypothes.is/api/annotations/](https://api.hypothes.is/api/annotations/K_JrMsbiEem5hutSGynvDw)”$1”
3. Save with ctrl o and exit with
4. Test bash DownloadNotes.sh K\_JrMsbiEem5hutSGynvDw

Error: DownloadNotes.sh: line 1: unexpected EOF while looking for matching `"'

DownloadNotes.sh: line 2: syntax error: unexpected end of file

Solution: I copy and pasted the command that I wrote in my learning journal. This error was solved by deleting the “$1” and writing it again in SWAN

Result: Success, annotations downloaded. I will delete this file (using rm) and write the rest of my script.

Objective: rename file

Action:

1. Type mv “$1” “$2”
2. Save with ctrl o, exit with ctrl x
3. Type bash DownloadNotes.sh K\_JrMsbiEem5hutSGynvDw programtest.txt
4. Type ls to test see if it was successful

Error: I ran into the same problem as last time.

Solution: I need to stop copy/pasting from my learning journal

Result: Success, the file is downloaded and renamed

Objective: Add comments

Action:

1. Type:

# Will download notes from hypothesis and rename them for you

# Usage: bash DownloadNotes.sh annotationID Filename

1. Give comments to my fiance and get him to run the program to see if they are understandable by someone else

Result: Success, although I had to tell him what the annotation ID is.

**Overall Result:** Success, I made my first script

**Overall Objective:** Delete some text automatically from the hypothesiscopy.txt file

Notes: My file currently looks like this:

{"id": "K\_JrMsbiEem5hutSGynvDw", "created": "2019-08-25T02:43:51.327033+00:00", "updated": "2019-08-30T08:46:31.519316+00:00", "user": "acct:georgiarutherford@hypothes.is", "uri": "https://plato.stanford.edu/entries/self-consciousness/", "text": "People to look at for my idea of the self and other: Fichte (1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a somewhat different perspective, Mead (1934; Aboulafia 1986)", "tags": ["Hegel", "self-consciousness", "self"], **"group": "\_\_world\_\_", "permissions": {"read": ["group:\_\_world\_\_"], "admin": ["acct:georgiarutherford@hypothes.is"], "update": ["acct:georgiarutherford@hypothes.is"], "delete": ["acct:georgiarutherford@hypothes.is"]}, "target": [{"source": "https://plato.stanford.edu/entries/self-consciousness/", "selector": [{"type": "RangeSelector", "endOffset": 1166, "startOffset": 835, "endContainer": "/div[1]/div[2]/div[3]/div[1]/div[1]/div[4]/p[11]", "startContainer": "/div[1]/div[2]/div[3]/div[1]/div[1]/div[4]/p[11]"}, {"end": 14815, "type": "TextPositionSelector", "start": 14484}, {"type": "TextQuoteSelector",** "exact": "Another, related tradition has argued that an awareness of subjects\nother than oneself is a necessary condition of self-consciousness (see\n \u00a74.4).\n Historical variations on such a view can be found in Fichte\n(1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a\nsomewhat different perspective, Mead (1934; Aboulafia 1986).", **"prefix": "n objective world (see\n \u00a74.3).\n ", "suffix": "\n\n\nFichte offers the most influe"}]}], "document": {"title": ["Self-Consciousness"]}, "links": {"html": "https://hypothes.is/a/K\_JrMsbiEem5hutSGynvDw", "incontext": "https://hyp.is/K\_JrMsbiEem5hutSGynvDw/plato.stanford.edu/entries/self-consciousness/", "json": "https://hypothes.is/api/annotations/K\_JrMsbiEem5hutSGynvDw"}, "flagged": false, "hidden": false, "user\_info": {"display\_name": null}}**

Ideally I would want my program to be able to delete all the text I have put in bold above and then separate out the rest of the text into sections (ID, Created, Updated, User, URL, Text, Tags, Exact). I will also want to keep the original text file for reference in case the program deletes something it shouldn’t. This will use a command input.txt > output.txt

My research on how to edit text file has led me to this <https://www.gnu.org/software/sed/manual/sed.html#Overview> .

Objective: Use sed command to edit my text file

Action:

1. Type sed ‘/”group”:/,/"TextQuoteSelector"/’ hypothesiscopy.txt > testdelete.txt

Error: File made with nothing in it. Might be because I forgot to use the d command

1. Type sed ‘/”group”:/,/"TextQuoteSelector"/d’ hypothesiscopy.txt > testdelete.txt

Error: bash: Type: command not found. Testdelete.txt made with nothing in it

1. Google some more because I am clearly missing something.
2. Read <https://unix.stackexchange.com/questions/243207/how-can-i-delete-everything-until-a-pattern-and-everything-after-another-pattern>
3. Type grep -o '"id".\*"group"' hypothesiscopy.txt > testdelete.txt
4. Type grep -o '"exact".\*"prefix"' hypothesiscopy.txt >> testdelete.txt

Result: Kind of a success, my testdelete.txt document now just shows:

"id": "K\_JrMsbiEem5hutSGynvDw", "created": "2019-08-25T02:43:51.327033+00:00", "updated": "2019-08-30T08:46:31.519316+00:00", "user": "acct:georgiarutherford@hypothes.is", "uri": "https://plato.stanford.edu/entries/self-consciousness/", "text": "People to look at for my idea of the self and other: Fichte (1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a somewhat different perspective, Mead (1934; Aboulafia 1986)", "tags": ["Hegel", "self-consciousness", "self"], "group"

"exact": "Another, related tradition has argued that an awareness of subjects\nother than oneself is a necessary condition of self-consciousness (see\n \u00a74.4).\n Historical variations on such a view can be found in Fichte\n(1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a\nsomewhat different perspective, Mead (1934; Aboulafia 1986).", "prefix"

This is the text I want to keep except the “group” and “prefix”, however I need them because for the code I need the words that will always be at the end of the text I want to copy. The text before them will change depending on the tags or quotes I use.

Objective: Delete “group” and “prefix” as well

Action:

1. Type grep -o '"id".\*"group"' | sed ‘“group”d’ hypothesiscopy.txt > testdelete.txt

Error: sed: -e expression #1, char 1: unknown command: `"'

Solution: the sed command needs to use ‘/^”group”d/ to indicate a word

1. Type grep -o '"id".\*"group"' | sed '/^"group"/d' hypothesiscopy.txt > testdelete.txt

Error: Didn’t do what I wanted. I had to cancel the task because it didn’t complete

Solution: I’m going to try this in a less complicated way and see if I can just delete “group” somehow

1. Type sed '/^"group"/d' hypothesiscopy.txt > testdelete.txt

Error: text still in text.

1. Google again: <https://www.cyberciti.biz/faq/howto-sed-exact-match-and-delete-words-within-a-file/>
2. Try sed 's/word-to-find//g' input.file > output.file
3. Type sed 's/”group”//g' hypthesiscopy.txt > testdelete.txt

Result: Success, the word group is gone

Objective: put some of these commands together

Action:

1. Type nano DownloadReformatTest.sh
2. Type:

# Will download notes from hypothesis, rename original, move info into a new file without unwanted sections

# Usage: bash DownloadReformatTest.sh annotationID Filename

wget [https://api.hypothes.is/api/annotations/"$1](https://api.hypothes.is/api/annotations/%22$1)"

grep -o '"id".\*"group"' “$1” > output.txt

sed 's/”group”//g' output.txt > “$2”

grep -o '"exact".\*"prefix"' “$1” > output.txt

sed 's/”prefix”//g' output.txt >> “$2”

rm “output.txt”

Mv “$1” “$2-original”

1. Test by typing bash DownloadReformatTest.sh K\_JrMsbiEem5hutSGynvDw testdeleteprogram.txt

Error: sed: -e expression #1, char 13: unterminated `s' command. Original document is saved, and testdeleteprogram.txt was made with the first section of text.

Solution: The error seems to be with line 5, it says: sed 's?"prefix"//g' output.txt >> "$2". I think I made a typo. Changing this to sed ‘s/ worked.

Error: the original copy is called testdeleteprogram.txt-original

Solution: change all the “$2” to “$2.txt” and the last one to “$2-original.txt”

Result: Success

**Overall Result:** Successfully made a script that downloads annotations from hypothes.is and delete a bunch of unwanted output. I doubt my script is the quickest way of doing this, but it is what made sense to my mind.

**Overall Objective:** Format further

Objective: Separate out my content using the grep command I learnt above

Action:

1. Type:

wget [https://api.hypothes.is/api/annotations/"$1](https://api.hypothes.is/api/annotations/%22$1)"

grep -o '"id".\*","' "$1" > “$2”.txt

grep -o '"created".\*","' "$1" >> “$2”.txt

grep -o '"updated".\*","' "$1" >> “$2”.txt

grep -o '"user".\*","' "$1" >> “$2”.txt

grep -o '"uri".\*","' "$1" >> “$2”.txt

grep -o '"text".\*","' "$1" >> “$2”.txt

grep -o '"tags".\*","' "$1" >> “$2”.txt

grep -o '"exact".\*","' "$1" >> “$2”.txt

mv “$1” “$2-original.txt”

1. Test by typing bash ReformatTest2.sh K\_JrMsbiEem5hutSGynvDw reformattest

Error: reformattest.txt has no text

Solution: I left in the “” from the last script, but that was apart of the actual word. Take these out

Error: The wild card \* means all, so it does this for all the , in the text. I want it to just do the next one. But even if I could do this I would run into the same problem I had using the csv. That is, commas are used in my quotations as well as separating categories

Solution: I am going to try to separate it by the words and then delete the first instance of every duplicated word.

1. First I will see if I can separate out by words by using the script

wget [https://api.hypothes.is/api/annotations/"$1](https://api.hypothes.is/api/annotations/%22$1)"

grep -o '"id".\*"created"' "$1" > “$2”.txt

grep -o '"created".\*"updated"' "$1" >> “$2”.txt

grep -o '"updated".\*"user"' "$1" >> “$2”.txt

grep -o '"user".\*"uri"' "$1" >> “$2”.txt

grep -o '"uri".\*"text"' "$1" >> “$2”.txt

grep -o '"text".\*"tags"' "$1" >> “$2”.txt

grep -o '"tags".\*"exact"' "$1" >> “$2”.txt

grep -o '"exact".\*"prefix"' "$1" >> “$2”.txt

mv “$1” “$2-original.txt”

1. Test by typing bash ReformatTest2.sh K\_JrMsbiEem5hutSGynvDw reformattest

Error: mv: cannot stat ‘“K\_JrMsbiEem5hutSGynvDw”’: No such file or directory

Solution: I hadn’t reformatted some of the “” after pasting them from my learning journal

1. Retype the “” and test by typing bash ReformatTest2.sh K\_JrMsbiEem5hutSGynvDw reformattest

Error: a lot of the info I don’t want is back in there

Solution: change step 8 to be from “tags” to “group” instead of “exact”

Result: Success, all the information is on a new line. However, each line also has an extra work at the end, such as "id": "K\_JrMsbiEem5hutSGynvDw", "created"

Objective: Remove the last word from each line

Action:

1. Type:

wget [https://api.hypothes.is/api/annotations/"$1](https://api.hypothes.is/api/annotations/%22$1)"

grep -o '"id".\*"created"' "$1" > output.txt

sed 's/"created"//g' output.txt > "$2".txt

grep -o '"created".\*"updated"' "$1" > output.txt

sed 's/"updated"//g' output.txt >> "$2".txt

grep -o '"updated".\*"user"' "$1" > output.txt

sed 's/"user"//g' output.txt >> "$2".txt

grep -o '"user".\*"uri"' "$1" > output.txt

sed 's/"uri"//g' output.txt >> "$2".txt

grep -o '"uri".\*"text"' "$1" > output.txt

sed 's/"text"//g' output.txt >> "$2".txt

grep -o '"text".\*"tags"' "$1" > output.txt

sed 's/"tags"//g' output.txt >> "$2".txt

grep -o '"tags".\*"group"' "$1" > output.txt

sed 's/"group"//g' output.txt >> "$2".txt

grep -o '"exact".\*"prefix"' "$1" > output.txt

sed 's/"prefix"//g' output.txt >> "$2".txt

rm output.txt

mv "$1" "$2-original.txt"

1. Test by typing bash ReformatTest2.sh K\_JrMsbiEem5hutSGynvDw reformattest

Result: Succes

Output in the reformattest-original.txt is:

{"id": "K\_JrMsbiEem5hutSGynvDw", "created": "2019-08-25T02:43:51.327033+00:00", "updated": "2019-08-30T08:46:31.519316+00:00", "user": "acct:georgiarutherford@hypothes.is", "uri": "https://plato.stanford.edu/entries/self-consciousness/", "text": "People to look at for my idea of the self and other: Fichte (1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a somewhat different perspective, Mead (1934; Aboulafia 1986)", "tags": ["Hegel", "self-consciousness", "self"], "group": "\_\_world\_\_", "permissions": {"read": ["group:\_\_world\_\_"], "admin": ["acct:georgiarutherford@hypothes.is"], "update": ["acct:georgiarutherford@hypothes.is"], "delete": ["acct:georgiarutherford@hypothes.is"]}, "target": [{"source": "https://plato.stanford.edu/entries/self-consciousness/", "selector": [{"type": "RangeSelector", "endOffset": 1166, "startOffset": 835, "endContainer": "/div[1]/div[2]/div[3]/div[1]/div[1]/div[4]/p[11]", "startContainer": "/div[1]/div[2]/div[3]/div[1]/div[1]/div[4]/p[11]"}, {"end": 14815, "type": "TextPositionSelector", "start": 14484}, {"type": "TextQuoteSelector", "exact": "Another, related tradition has argued that an awareness of subjects\nother than oneself is a necessary condition of self-consciousness (see\n \u00a74.4).\n Historical variations on such a view can be found in Fichte\n(1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a\nsomewhat different perspective, Mead (1934; Aboulafia 1986).", "prefix": "n objective world (see\n \u00a74.3).\n ", "suffix": "\n\n\nFichte offers the most influe"}]}], "document": {"title": ["Self-Consciousness"]}, "links": {"html": "https://hypothes.is/a/K\_JrMsbiEem5hutSGynvDw", "incontext": "https://hyp.is/K\_JrMsbiEem5hutSGynvDw/plato.stanford.edu/entries/self-consciousness/", "json": "https://hypothes.is/api/annotations/K\_JrMsbiEem5hutSGynvDw"}, "flagged": false, "hidden": false, "user\_info": {"display\_name": null}}

Output in the reformattest.txt is:

"id": "K\_JrMsbiEem5hutSGynvDw",

"created": "2019-08-25T02:43:51.327033+00:00",

"updated": "2019-08-30T08:46:31.519316+00:00",

"user": "acct:georgiarutherford@hypothes.is",

"uri": "https://plato.stanford.edu/entries/self-consciousness/",

"text": "People to look at for my idea of the self and other: Fichte (1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a somewhat different perspective, Mead (1934; Aboulafia 1986)",

"tags": ["Hegel", "self-consciousness", "self"],

"exact": "Another, related tradition has argued that an awareness of subjects\nother than oneself is a necessary condition of self-consciousness (see\n \u00a74.4).\n Historical variations on such a view can be found in Fichte\n(1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a\nsomewhat different perspective, Mead (1934; Aboulafia 1986).",

**Overall result:** I am very happy with my results so far. I will have to be very careful about using individual words in quotations as if the word is id, created, updated, user, uri, text, tags, exact, group or prefix it would mess up my code. I do not foresee this becoming too much of an issue, but I have made sure my code keeps a copy of the original document just in case.

**Week 6**

**Overall Objective:** Write up my Design document in LaTex

Action: write up my design using headings, dot points and numbered lists

Error: For some reason every time I click enter to make a new line it comes up as a numbered point

Solution: By checking the source I can see that somehow I have the \begin{enumerate} but not the \end{enumerate}. I must have accidentally deleted it. This is fixed by writing \end{enumerate} where I want the numbered list to end.

**Overall Result:** Success

**Overall Objective:** Make and export report of project management

Objective: Make Project in Github

Action:

1. Click on Projects
2. Click on New Project
3. Name Project Board
4. Add Description
5. Choose project template Basic Kanban (which I have started using for my normal to do lists and is great)
6. Click create project
7. Add column called Backlog
8. Move Backlog to behind To Do
9. Click the plus above the column to create cards
10. Add in User Stories as cards
11. Shift 5 cards into To Do
12. Shift 3 of those cards into In Progress

Result: Success

Objective: Export report to post on Cloudstor

Action:

1. Google how to export a report from your project github
2. Find literally no information on how to do this
3. Screenshot the page to post of cloudstor

Result: Kind of a success, the information will be on cloudstor at least

**Overall Result**: Success

**Week 7**

**Overall Objective:** Save multiple annotations from the one web page

Objective: Make some more annotations on the test page

Action:

1. Go to <https://plato.stanford.edu/entries/self-consciousness/#annotations:K_JrMsbiEem5hutSGynvDw>
2. Turn on hypothes.is
3. Make notes
4. Post as public

Result: Success

Objective: Use previously made code to get multiple annotations

Note: I suspect there may be issues with this due to multiple instances of “created”, “id” ect.

Action:

1. Open SWAN

Error: The files I was working on before don’t appear to be where they were last time

Solution: It appears that SWAN saves previous sessions within the cloudstor directory, within the SWAN\_Sessions directory and within a directory titled with the date it was done on

1. Open previous session

Note: I decided to leave this session alone so that I have a record of what I achieved on this day. I’ll start a new Session for today's work.

1. Copy the Code from ReformatTest2.sh
2. Move back to the root directory
3. Type mkdir Notes
4. Move into notes directory
5. Type nano ReformatTest.sh
6. Paste code into new file
7. Change Instruction # Usage: bash ReformatTest2.sh annotationID Filename to # Usage: bash ReformatTest.sh annotationID Filename
8. Save and Exit
9. Go to hypothes.is
10. Click “visit annotations in context” and get the annotation ID for the web page
11. Type bash ReformatTest.sh lSj-ltXDEemEHhOTFjVDhA MultipleTest

Error: As suspected the Reformatted file only contains information for one annotation

Error: Unexpectedly the original file also only contains information for one annotation. It seems that “visit annotations in context” takes you to the URL for the last annotation made on that web page.

Objective: Figure out how to get multiple annotations

Action:

1. Google hypothes.is using api to fetch more than one annotation
2. Find <https://web.hypothes.is/blog/using-the-new-wildcard-url-search-api-to-monitor-sitewide-annotation-activity/> (about using a wildcard url to search sitewide for annotations, doesn’t seem exactly right, but maybe it could help)
3. Type wget [https://hypothes.is/search?q=url:https://plato.stanford.edu/entries/self-consciousness/\*](https://hypothes.is/search?q=url:https://plato.stanford.edu/entries/self-consciousness/*)

Result: It definitely did something, I got a lot of text in a file that is not very clear

1. In web page type [https://hypothes.is/search?q=url:https://plato.stanford.edu/entries/self-consciousness/\*](https://hypothes.is/search?q=url:https://plato.stanford.edu/entries/self-consciousness/*)

Result: This worked to search the hypothes.is website for the annotations on the page. Most of the text I was getting back must have been the formatting of the website

1. Look up the API for search on hypothes.is here: <https://h.readthedocs.io/en/latest/api-reference/v1/#tag/annotations/paths/~1search/get>
2. Try https://api.hypothes.is/api/search/q=url:https://plato.stanford.edu/entries/self-consciousness/\*

Error: Failure to get any information

1. Try [https://api.hypothes.is/api/search?q=url:https://plato.stanford.edu/entries/self-consciousness/\*](https://api.hypothes.is/api/search?q=url:https://plato.stanford.edu/entries/self-consciousness/*)

Error: This got lots of information. Unfortunately none of it appears to be from the actual website I was trying to search. Searching the information for uri shows that some of these annotations are from google docs and some are from wordpress sites, so I don’t quite understand what it was trying to search. If I search the text there aren’t even any references to the word ‘self’ in any of these annotations. I am especially confused because the non-API version of this seemed to work just fine.

Solution: After a lot of fiddling I have realised that the above is just getting the same output as <https://hypothes.is/search>? Does on the website, so the api is just getting me to the search section. Still not sure how to fix this exactly.

1. Continue googling
2. Find <https://web.hypothes.is/blog/viewing-and-exporting-hypothesis-annotations/> which mentions searching and says The underlying API query in this case, <https://hypothes.is/api/search?any=candelalearning>
3. Try <https://hypothes.is/api/search?any=https://plato.stanford.edu/entries/self-consciousness/>

Result: Almost a success, all my annotations are there but there are also annotations from other websites which are very similar

1. Try <https://hypothes.is/api/search?url=https://plato.stanford.edu/entries/self-consciousness/>

Result: Success! The change from search?any= to search?url= meant it only searched for annotations on that specific url. All my annotations are there and in the format I am used to from using the <https://api.hypothes.is/api/annotations/> command. This is good because I should still be able to use the code I have previously been working on.

1. Type wget <https://hypothes.is/api/search?url=https://plato.stanford.edu/entries/self-consciousness/> into SWAN terminal

Result: Success! I got the information from the API on terminal.

Objective: Rework the code so that it works with the new API code

Action:

1. Type nano ReformatTest.sh
2. Change note to say # Usage: bash ReformatTest.sh **URL** Filename
3. Change wget https://api.hypothes.is/api/annotations/"$1" to wget <https://hypothes.is/api/search?url=>“$1”
4. Change all other “$1” to search?url="$1"

# Will download notes from hypothesis, rename original, move info into a new file without unwanted sections

# Usage: bash ReformatTest.sh URL Filename

wget https://hypothes.is/api/search?url="$1"

grep -o '"id".\*"created"' search?url="$1" > output.txt

sed 's/"created"//g' output.txt > "$2".txt

grep -o '"created".\*"updated"' search?url="$1" > output.txt

sed 's/"updated"//g' output.txt >> "$2".txt

grep -o '"updated".\*"user"' search?url="$1" > output.txt

sed 's/"user"//g' output.txt >> "$2".txt

grep -o '"user".\*"uri"' search?url="$1" > output.txt

sed 's/"uri"//g' output.txt >> "$2".txt

grep -o '"uri".\*"text"' search?url="$1" > output.txt

sed 's/"text"//g' output.txt >> "$2".txt

grep -o '"text".\*"tags"' search?url="$1" > output.txt

sed 's/"tags"//g' output.txt >> "$2".txt

grep -o '"tags".\*"group"' search?url="$1" > output.txt

sed 's/"group"//g' output.txt >> "$2".txt

grep -o '"exact".\*"prefix"' search?url="$1" > output.txt

sed 's/"prefix"//g' output.txt >> "$2".txt

rm output.txt

mv search?url="$1" "$2-original.txt"

1. Test by typing bash ReformatTest.sh <https://plato.stanford.edu/entries/self-consciousness/> MultipleTest2

Error: Did not work, MultipleTest2.txt has no text, the file did not rename. I think the problem is that that when it saved the file was named so that all the / are %2F. I need to be able to rename the file without knowing what the previous name is as the slashes are going to be different in every URL.

1. Type wget <https://hypothes.is/api/search?url=https://plato.stanford.edu/entries/self-consciousness/> | mv search?url\* test

Result: Success, file renamed. My thought process here is that I’m never going to name another document anything starting with search?url, so I can rename all files starting with that. Since every time I run this program the file will be automatically renamed and I am never going to use that naming style, it should work that only the new file is renamed.

1. Change the code (adding in a warning that it will rename files, renaming the file at the start from the url to original)

# Will download notes from hypothesis, rename original, move info into a new file without unwanted sections

#Warning: This will rename all files beginning with search?url

# Usage: bash ReformatTest.sh URL Filename

wget https://hypothes.is/api/search?url="$1"

mv search?url original

grep -o '"id".\*"created"' original > output.txt

sed 's/"created"//g' output.txt > "$2".txt

grep -o '"created".\*"updated"' original > output.txt

sed 's/"updated"//g' output.txt >> "$2".txt

grep -o '"updated".\*"user"' original > output.txt

sed 's/"user"//g' output.txt >> "$2".txt

grep -o '"user".\*"uri"' original > output.txt

sed 's/"uri"//g' output.txt >> "$2".txt

grep -o '"uri".\*"text"' original > output.txt

sed 's/"text"//g' output.txt >> "$2".txt

grep -o '"text".\*"tags"' original > output.txt

sed 's/"tags"//g' output.txt >> "$2".txt

grep -o '"tags".\*"group"' original > output.txt

sed 's/"group"//g' output.txt >> "$2".txt

grep -o '"exact".\*"prefix"' original > output.txt

sed 's/"prefix"//g' output.txt >> "$2".txt

rm output.txt

mv original "$2-original.txt"

1. Test code by typing bash ReformatTest.sh https://plato.stanford.edu/entries/self-consciousness/ MultipleTest3

Error: mv: cannot stat ‘search?url’: No such file or directory

Solution: I forgot the wildcard \*

Result: I now have the two files I want (MultipleTest3.txt and MultipleTest3-original.txt) and these are separated out on lines based on the categories of ID, User, Updated ect. However, as previously predicted I have ran into some more issues now that the text is working on multiple annotations. For now I will say that I am able to successfully save the information of multiple annotations. However, my next issue to work on is cleaning this annotation information so that I have something clearly understandable by humans.

**Overall Result:** Successfully saved multiple annotations

**Overall Objective:** Clean multiple annotations so that they are human readable

Objective: Change the code so that it only works for the first instance of a word

Note: I am hoping that if I learn how to change only the first instance of a word it might make it easier to understand how to then do the others afterwards

Action:

1. Google terminal change the first instance of a word
2. Found this on sed commands <https://unix.stackexchange.com/questions/155805/sed-replace-first-k-instances-of-a-word-in-the-file> and this on grep commands <https://unix.stackexchange.com/questions/358523/find-only-the-first-occurence-using-only-grep>
3. Type nano ReformatMultipleTest.sh to make a new file
4. Type the following:

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original

grep -o -m 1'"id".\*"created"' original | head -1 > output.txt

sed 's/"created"//1' output.txt > "$2".txt

grep -o -m 1 '"created".\*"updated"' original | head -1 > output.txt

sed 's/"updated"//1' output.txt >> "$2".txt

grep -o -m 1 '"updated".\*"user"' original | head -1 > output.txt

sed 's/"user"//1' output.txt >> "$2".txt

grep -o -m 1 '"user".\*"uri"' original | head -1 > output.txt

sed 's/"uri"//1' output.txt >> "$2".txt

grep -o -m 1 '"uri".\*"text"' original | head -1> output.txt

sed 's/"text"//1' output.txt >> "$2".txt

grep -o -m 1 '"text".\*"tags"' original | head -1> output.txt

sed 's/"tags"//1' output.txt >> "$2".txt

grep -o -m 1 '"tags".\*"group"' original | head -1> output.txt

sed 's/"group"//1' output.txt >> "$2".txt

grep -o -m 1'"exact".\*"prefix"' original | head -1> output.txt

sed 's/"prefix"//1' output.txt >> "$2".txt

rm output.txt

mv original "$2-original.txt"

1. Test by typing bash ReformatMultipleTest.sh https://plato.stanford.edu/entries/self-consciousness/ MultipleTest4

Error: grep: invalid max count. I also realised that as I am moving the text over to the output.txt file I don’t need to change the sed command. It's just the grep command that I need to change.

1. Try get rid of the -m 1

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original

grep -o '"id".\*null}}' original | head -1 > output.txt

Result: Still getting text between the first ID and the last null

1. Google again: <https://unix.stackexchange.com/questions/180663/how-to-select-first-occurrence-between-two-patterns-including-them>
2. Based on that try awk '/P1/,/P2/ { print } /P2/ { exit }', so awk '/"id"/,/null}}/ { print } /null}}/ { exit }'

Error: no text, I realised I’m not saying what file to do this on. I think I’m missing a step on how to use awk.

1. Try

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

cat original.txt | awk '/"id"/,/null}}/ { print }' original.txt > output.txt

Error: Same text in output.txt as in original.txt, no change

1. Try

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

cat original.txt | awk '/"id"/,/null}}/ { print }' > output.txt

Error: Same text in output.txt as in original.txt. Looking at it closer I think this might be because all my information is on one line

1. Google again find <https://www.unix.com/solaris/25124-how-i-can-use-word-seperator.html>
2. Try cat original.txt | awk -v FS=’"id":’ '{print $2}' > output.txt

Error: this worked better but it only printed one of the id section. I think it only printed the second one because of the {print $2}

1. Try cat original.txt | awk -v FS=’"id":’ '{print $3}' > output.txt

Result: I was correct changing to {print$3} changed the section that was output.

1. Try to do the above task as code

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

cat original.txt | awk -v FS=’"id":’ '{print $1}' > output1.txt

Error: {print $1} is actually whats before the first”id”: which is not what I want. I want to start processing from {print $2}

Solution: change {print $1} to {print $2}

1. Try put it all together using the following code

# Will download notes from hypothesis, rename original, move info into a new file without unwanted sections

# Warning: This will rename all files beginning with search?url

# Usage: bash ReformatMultipleTest.sh URL Filename

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

cat original.txt | awk -v FS=’"id":’ '{print $2}' > output1.txt

grep -o '"created".\*"updated"' output1.txt > output2.txt

sed 's/"updated"//g' output2.txt >> "$2".txt

grep -o '"updated".\*"user"' output1.txt > output2.txt

sed 's/"user"//g' output2.txt >> "$2".txt

grep -o '"user".\*"uri"' output1.txt > output2.txt

sed 's/"uri"//g' output2.txt >> "$2".txt

grep -o '"uri".\*"text"' output1.txt > output2.txt

sed 's/"text"//g' output2.txt >> "$2".txt

grep -o '"text".\*"tags"' output1.txt > output2.txt

sed 's/"tags"//g' output2.txt >> "$2".txt

grep -o '"tags".\*"group"' output1.txt > output2.txt

sed 's/"group"//g' output2.txt >> "$2".txt

grep -o '"exact".\*"prefix"' output1.txt > output2.txt

sed 's/"prefix"//g' output2.txt >> "$2".txt

rm output1.txt

rm output2.txt

mv original.txt "$2-original.txt"

1. Test by typing bash ReformatMultipleTest.sh https://plato.stanford.edu/entries/self-consciousness/ MultipleTest5

Error: no text in MultipleTest5

Solution: Once again I copy/pasted from my learning journal and some of the ‘’ needed to be formatted

Result: Success! The first of my annotations is formatted in the correct way (shown below). Now I just need to find a way of looping this so that they are all formatted in this way

"created": "2019-09-13T01:12:40.864229+00:00",

"updated": "2019-09-13T01:12:40.864229+00:00",

"user": "acct:georgiarutherford@hypothes.is",

"uri": "https://plato.stanford.edu/entries/self-consciousness/",

"text": "This is where the asymmetry of self and others comes in",

"tags": ["asymmetry", "self-consciousness"],

"exact": "Since knowledge of other minds is\ntypically considered to be open to sceptical doubt, and\nself-consciousness is not, such lines of reasoning are transcendental\narguments and so potentially open to general criticisms of that form\nof argument ",

**Overall Result:** So far I have managed to take a file that has the information for multiple annotations and separate out the information for one of those annotations. Although I have not yet managed this for multiple annotations, I am going to stop for today and come back to it later.

**Overall Objective:** Separate out the information for multiple annotations

Objective: Set up a new SWAN session

Action:

1. Type mkdir Notes
2. Type cd Notes

Result: Success

Objective: Loop my code changing the {print $2} section each time

Action:

1. Google how to loop while changing one variable in bash
2. Find <https://www.cyberciti.biz/faq/bash-for-loop/> and <https://stackoverflow.com/questions/1521462/looping-through-the-content-of-a-file-in-bash>
3. Try the following code:

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

cat original.txt | For i in {1, 2, 3}

do

awk -v FS='"id":' '{print $i}' > output1.txt

grep -o '"created".\*"updated"' output1.txt > output2.txt

sed 's/"updated"//g' output2.txt >> "$2".txt

grep -o '"updated".\*"user"' output1.txt > output2.txt

sed 's/"user"//g' output2.txt >> "$2".txt

grep -o '"user".\*"uri"' output1.txt > output2.txt

sed 's/"uri"//g' output2.txt >> "$2".txt

grep -o '"uri".\*"text"' output1.txt > output2.txt

sed 's/"text"//g' output2.txt >> "$2".txt

grep -o '"text".\*"tags"' output1.txt > output2.txt

sed 's/"tags"//g' output2.txt >> "$2".txt

grep -o '"tags".\*"group"' output1.txt > output2.txt

sed 's/"group"//g' output2.txt >> "$2".txt

grep -o '"exact".\*"prefix"' output1.txt > output2.txt

sed 's/"prefix"//g' output2.txt >> "$2".txt

done

rm output1.txt

rm output2.txt

mv original.txt "$2-original.txt"

Error: For: command not found, cat: write error: Broken pipe, Test.sh: line 8: syntax error near unexpected token `do', Test.sh: line 8: `do'

Solution: I wrote the for command with a capital letter, write it in lowercase and the code will run

Error: It grabbed way too much text

1. Try code again but this time just with for i in {1} do

Error: Also grabbed way too much text, so I clearly don’t understand what the for command is changing.

Result: This clearly hasn’t worked, but I want to take it back a couple steps and first see if what I want to loop actually works the way I think it should

Objective: Get the first, second and third annotations by using one code but without looping

Action:

1. Type nano LongTest.sh
2. Type the following incredibly long code (the {print$2} goes up by 1 each time)

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

cat original.txt | awk -v FS='"id":' '{print $2}' > output1.txt

grep -o '"created".\*"updated"' output1.txt > output2.txt

sed 's/"updated"//g' output2.txt >> "$2".txt

grep -o '"updated".\*"user"' output1.txt > output2.txt

sed 's/"user"//g' output2.txt >> "$2".txt

grep -o '"user".\*"uri"' output1.txt > output2.txt

sed 's/"uri"//g' output2.txt >> "$2".txt

grep -o '"uri".\*"text"' output1.txt > output2.txt

sed 's/"text"//g' output2.txt >> "$2".txt

grep -o '"text".\*"tags"' output1.txt > output2.txt

sed 's/"tags"//g' output2.txt >> "$2".txt

grep -o '"tags".\*"group"' output1.txt > output2.txt

sed 's/"group"//g' output2.txt >> "$2".txt

grep -o '"exact".\*"prefix"' output1.txt > output2.txt

sed 's/"prefix"//g' output2.txt >> "$2".txt

cat original.txt | awk -v FS='"id":' '{print $3}' > output1.txt

grep -o '"created".\*"updated"' output1.txt > output2.txt

sed 's/"updated"//g' output2.txt >> "$2".txt

grep -o '"updated".\*"user"' output1.txt > output2.txt

sed 's/"user"//g' output2.txt >> "$2".txt

grep -o '"user".\*"uri"' output1.txt > output2.txt

sed 's/"uri"//g' output2.txt >> "$2".txt

grep -o '"uri".\*"text"' output1.txt > output2.txt

sed 's/"text"//g' output2.txt >> "$2".txt

grep -o '"text".\*"tags"' output1.txt > output2.txt

sed 's/"tags"//g' output2.txt >> "$2".txt

grep -o '"tags".\*"group"' output1.txt > output2.txt

sed 's/"group"//g' output2.txt >> "$2".txt

grep -o '"exact".\*"prefix"' output1.txt > output2.txt

sed 's/"prefix"//g' output2.txt >> "$2".txt

cat original.txt | awk -v FS='"id":' '{print $4}' > output1.txt

grep -o '"created".\*"updated"' output1.txt > output2.txt

sed 's/"updated"//g' output2.txt >> "$2".txt

grep -o '"updated".\*"user"' output1.txt > output2.txt

sed 's/"user"//g' output2.txt >> "$2".txt

grep -o '"user".\*"uri"' output1.txt > output2.txt

sed 's/"uri"//g' output2.txt >> "$2".txt

grep -o '"uri".\*"text"' output1.txt > output2.txt

sed 's/"text"//g' output2.txt >> "$2".txt

grep -o '"text".\*"tags"' output1.txt > output2.txt

sed 's/"tags"//g' output2.txt >> "$2".txt

grep -o '"tags".\*"group"' output1.txt > output2.txt

sed 's/"group"//g' output2.txt >> "$2".txt

grep -o '"exact".\*"prefix"' output1.txt > output2.txt

sed 's/"prefix"//g' output2.txt >> "$2".txt

rm output1.txt

rm output2.txt

mv original.txt "$2-original.txt"

1. Run by typing bash LongTest.sh https://plato.stanford.edu/entries/self-consciousness/ MultipleTest2

Result: Success, the three annotations are formatted exactly how I expected them to be

Objective: Put some echo commands into the script

Note: As my code is getting more complicated I have find that I want to know what step the code is up to, so I want to add in an echo so that it will tell me.

Action:

1. Edit code to look like this

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

cat original.txt | awk -v FS='"id":' '{print $2}' > output1.txt

grep -o '"created".\*"updated"' output1.txt > output2.txt

sed 's/"updated"//g' output2.txt >> "$2".txt

echo "created done"

grep -o '"updated".\*"user"' output1.txt > output2.txt

sed 's/"user"//g' output2.txt >> "$2".txt

echo "updated done"

grep -o '"user".\*"uri"' output1.txt > output2.txt

sed 's/"uri"//g' output2.txt >> "$2".txt

echo "user done"

grep -o '"uri".\*"text"' output1.txt > output2.txt

sed 's/"text"//g' output2.txt >> "$2".txt

echo "URL done"

grep -o '"text".\*"tags"' output1.txt > output2.txt

sed 's/"tags"//g' output2.txt >> "$2".txt

echo "text done"

grep -o '"tags".\*"group"' output1.txt > output2.txt

sed 's/"group"//g' output2.txt >> "$2".txt

echo "tags done"

grep -o '"exact".\*"prefix"' output1.txt > output2.txt

sed 's/"prefix"//g' output2.txt >> "$2".txt

echo "exact done"

rm output1.txt

rm output2.txt

mv original.txt "$2-original.txt"

echo "task complete"

1. Test by typing bash Test.sh https://plato.stanford.edu/entries/self-consciousness/ MultipleTest3

Result: Success. I may change where the echo is placed in the future, but this works for now.

Objective: Get the information of the three different annotations

Action:

1. Google how to loop a code
2. Find nothing that seems to fit
3. Google how to change a variable in a code
4. Find <https://linuxize.com/post/bash-increment-decrement-variable/> which seems promising
5. Edit my code to include an until loop and the incremental increase so that the code looks like:

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

i=1

until [ $i -gt 4 ]

do

cat original.txt | awk -v FS='"id":' '{print $'$i'}' > output1.txt

grep -o '"created".\*"updated"' output1.txt > output2.txt

sed 's/"updated"//g' output2.txt >> "$2".txt

echo "created done"

grep -o '"updated".\*"user"' output1.txt > output2.txt

sed 's/"user"//g' output2.txt >> "$2".txt

echo "updated done"

grep -o '"user".\*"uri"' output1.txt > output2.txt

sed 's/"uri"//g' output2.txt >> "$2".txt

echo "user done"

grep -o '"uri".\*"text"' output1.txt > output2.txt

sed 's/"text"//g' output2.txt >> "$2".txt

echo "URL done"

grep -o '"text".\*"tags"' output1.txt > output2.txt

sed 's/"tags"//g' output2.txt >> "$2".txt

echo "text done"

grep -o '"tags".\*"group"' output1.txt > output2.txt

sed 's/"group"//g' output2.txt >> "$2".txt

echo "tags done"

grep -o '"exact".\*"prefix"' output1.txt > output2.txt

sed 's/"prefix"//g' output2.txt >> "$2".txt

echo "exact done"

((i++))

done

rm output1.txt

rm output2.txt

mv original.txt "$2-original.txt"

echo "task complete"

Result: Success! The document has the information I want all separated out into the sections I want.

**Overall Result:** Success, I have the information of three annotations formatted. My next step is to change the loop so that it continues until there is no more text to format, rather than just a set number of times.

**Overall Objective:** Make the loop be able to repeat for as many annotations I have done

Objective: Test using grep to count how many words are in a document

Action:

1. Google using grep to count the instances of a word in a document
2. Find <https://www.cyberciti.biz/faq/how-to-count-total-number-of-word-occurrences-using-grep-on-linux-or-unix/> which says the command I wand is grep -c string filename
3. Try grep -c "created" MultipleTest1-original.txt

Error: It said there was one instance when there were actually three. I’m going to take it back a couple steps and test it on a less complicated document

1. Type nano testgrep.txt
2. Type hello hello hello and then one more hello on a new line
3. Save and exit
4. In terminal type grep -c hello testgrep.txt

Error: it said there were two instances of hello. Although the instructions I was following said it counts the total instances of the word, it must be counting how many lines have an instance of hello rather than every instance. I think I could use this with a pipeline and the grep -o command I have previously used.

1. Type grep -o hello testgrep.txt | grep -c hello

Result: Success, it says there are four instances of grep. Now I want to try this one of my files containing annotation information.

1. Type grep -o "created" MultipleTest1-original.txt | grep -c "created"

Result: Success, it says there are 3 instances of “created”

Objective: Change a variable in my code based on the count it receives from a count

Action:

1. Google how to change a variable in a code based on a count
2. Find <https://stackoverflow.com/questions/16327355/assign-grep-count-to-variable>
3. Try adding var=$(grep -o "created" original.txt | grep -c "created") and changing the number to var
4. Test by typing bash Test.sh https://plato.stanford.edu/entries/self-consciousness/ MultipleTest5

Error: It kept looping, I could tell because of the echo command I put into the code. The error was line 10: [: var: integer expression expected.

1. Try changing the var to $var

Error: Changing var to $var was successful, but since there were three instances of the word “created” the variable was changed to 3. Unfortunately the text before the first “created” counts as a run through the code, so I need to add one to the variable.

1. Try changing it to until [ $i -gt $var + 1 ]

Error: received error too many arguments

1. Try getting rid of the spaces

Error: 3+1: integer expression expected

1. Try adding in a second variable var2=$(($var+1))

Result: Success. My code determines how many instances of the word “created” that are in the document, adds one to that amount and then runs through the code that many times. It looks like:

# Will download notes from hypothesis, rename original, move info into a new file without unwanted sections

# Warning: This will rename all files beginning with search?url

# Usage: bash Test.sh URL Filename

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

var=$(grep -o "created" original.txt | grep -c "created")

var2=$(($var+1))

echo variable found

i=1

until [ $i -gt $var2 ]

do

cat original.txt | awk -v FS='"id":' '{print $'$i'}' > output1.txt

grep -o '"created".\*"updated"' output1.txt > output2.txt

sed 's/"updated"//g' output2.txt >> "$2".txt

echo "created done"

grep -o '"updated".\*"user"' output1.txt > output2.txt

sed 's/"user"//g' output2.txt >> "$2".txt

echo "updated done"

grep -o '"user".\*"uri"' output1.txt > output2.txt

sed 's/"uri"//g' output2.txt >> "$2".txt

echo "user done"

grep -o '"uri".\*"text"' output1.txt > output2.txt

sed 's/"text"//g' output2.txt >> "$2".txt

echo "URL done"

grep -o '"text".\*"tags"' output1.txt > output2.txt

sed 's/"tags"//g' output2.txt >> "$2".txt

echo "text done"

grep -o '"tags".\*"group"' output1.txt > output2.txt

sed 's/"group"//g' output2.txt >> "$2".txt

echo "tags done"

grep -o '"exact".\*"prefix"' output1.txt > output2.txt

sed 's/"prefix"//g' output2.txt >> "$2".txt

echo "exact done"

((i++))

done

rm output1.txt

rm output2.txt

mv original.txt "$2-original.txt"

echo "task complete"

**Overall Result:** Success. My next step is to format my annotation information further and to start analysing the information.

**Overall Objective**: Improve the readability

Objective: Try put into spreadsheet format

Note: I know the standard .csv style of separating by commas won’t work as my notes somehow have commas, so first I will want to change the “: of “created”: to something else that I can separate with. As I am unlikely to use the | in my note taking, I will use this.

Action:

1. Type sed 's/”:/|/g' MultipleTest6.txt

Error: this worked, but I will want to move this into a file of some kind so that I can save it

1. Type sed 's/":/|/g' MultipleTest6.csv > FormatTest.txt

Error: This worked, but I will probably want spaces between the first word and the pipe

1. Type sed 's/":/ |/g' MultipleTest6.csv > FormatTest.txt

Result: This worked and I think it's clearer. I would also be able to put this into csv format and change to separated by |. I’m not sure if I can do this step in terminal, but I was able to open the FormatTest.csv file using the gui interface and do this successfully. I’m still not completely sure if I prefer using the .csv though, so for now I will continue cleaning with it as a .txt file.

Objective: Remove some of the extra characters

Action:

1. Use sed 's/"//' FormatTest.txt to remove only the first instance of “
2. Try 's/"//' FormatTest.txt > FormatTest.txt

Error: for some reason I can’t move from a document into the same document. The result is that I get an empty document.

Solution: If I use the command 's/"//' FormatTest.txt > FormatTest1.txt it is successful. For some reason it only works if I move it into a new file.

1. Google how to remove the last character of a line find <http://www.theunixschool.com/2014/08/sed-examples-remove-delete-chars-from-line-file.html> which has the code for deleting the first and last character of a line by using ; to connect them.
2. sed 's/.//;s/.$//' FormatTest.txt

Error: The last character is still there. If I make a test document with a few lines of “hello” the code works. It must have issues deleting the comma that is at the end of my lines.

Solution: After a lot of fiddling with it, googling, getting confused, and looking back and forth at the file I realised that the comma isn’t actually the last character, there is actually a space after the comma. So the command sed 's/ $//;s/,$//' FormatTest1.txt worked to get rid of the space and then the comma.

Result: At this stage this is all the extra characters I would like to get rid of. At some stage I might think about getting rid of some more of the “” or [] but this is more difficult as some of the lines are formatted differently so I couldn’t apply it uniformly across every line.

Objective: Separate out each annotation so that I can easily see when a new annotation begins.

Action:

1. Google bash add a blank line after every 7 lines
2. Find <https://unix.stackexchange.com/questions/23545/insert-a-new-line-after-every-n-lines>
3. Try sed '0~7 a\\' FormatTest1.txt

Result: Success

Objective: Put all the things I’ve been working on into my code

Action:

1. Create a new file by typing nano FormatTest.sh
2. Write the following code:

# Will download notes from hypothesis, rename original, move info into a new file without unwanted sections

# Warning: This will rename all files beginning with search?url

# Usage: bash FormatTest.sh URL Filename

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

var=$(grep -o "created" original.txt | grep -c "created")

var2=$(($var+1))

i=1

until [ $i -gt $var2 ]

do

cat original.txt | awk -v FS='"id":' '{print $'$i'}' > output1.txt

grep -o '"created".\*"updated"' output1.txt > output2.txt

sed 's/"updated"//g' output2.txt >> output3.txt

grep -o '"updated".\*"user"' output1.txt > output2.txt

sed 's/"user"//g' output2.txt >> output3.txt

grep -o '"user".\*"uri"' output1.txt > output2.txt

sed 's/"uri"//g' output2.txt >> output3.txt

grep -o '"uri".\*"text"' output1.txt > output2.txt

sed 's/"text"//g' output2.txt >> output3.txt

grep -o '"text".\*"tags"' output1.txt > output2.txt

sed 's/"tags"//g' output2.txt >> output3.txt

grep -o '"tags".\*"group"' output1.txt > output2.txt

sed 's/"group"//g' output2.txt >> output3.txt

grep -o '"exact".\*"prefix"' output1.txt > output2.txt

sed 's/"prefix"//g' output2.txt >> output3.txt

((i++))

echo "An annotation complete"

done

sed 's/":/ |/g;s/.//;s/ $//;s/,$//' output3.txt > output2.txt

sed '0~7 a\\' output2.txt > "$2".txt

rm output1.txt

rm output2.txt

rm output3.txt

mv original.txt "$2-original.txt"

echo "task complete"

1. Test by typing bash FormatTest.sh https://plato.stanford.edu/entries/self-consciousness/ FormatTest2

Result: Output below

created | "2019-09-13T01:12:40.864229+00:00"

updated | "2019-09-13T01:12:40.864229+00:00"

user | "acct:georgiarutherford@hypothes.is"

uri | "https://plato.stanford.edu/entries/self-consciousness/"

text | "This is where the asymmetry of self and others comes in"

tags | ["asymmetry", "self-consciousness"]

exact | "Since knowledge of other minds is\ntypically considered to be open to sceptical doubt, and\nself-consciousness is not, such lines of reasoning are transcendental\narguments and so potentially open to general criticisms of that form\nof argument "

created | "2019-09-13T01:09:48.328890+00:00"

updated | "2019-09-13T01:09:48.328890+00:00"

user | "acct:georgiarutherford@hypothes.is"

uri | "https://plato.stanford.edu/entries/self-consciousness/"

text | "This is what I want to argue in my thesis"

tags | ["self-consciousness"]

exact | "is to argue that knowledge of other minds is a necessary\ncondition of the possibility of self-consciousness"

created | "2019-08-25T02:43:51.327033+00:00"

updated | "2019-08-30T08:46:31.519316+00:00"

user | "acct:georgiarutherford@hypothes.is"

uri | "https://plato.stanford.edu/entries/self-consciousness/"

text | "People to look at for my idea of the self and other: Fichte (1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a somewhat different perspective, Mead (1934; Aboulafia 1986)"

tags | ["Hegel", "self-consciousness", "self"]

exact | "Another, related tradition has argued that an awareness of subjects\nother than oneself is a necessary condition of self-consciousness (see\n \u00a74.4).\n Historical variations on such a view can be found in Fichte\n(1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a\nsomewhat different perspective, Mead (1934; Aboulafia 1986)."

**Overall Result:** I think this session was very successful, the Information is much clearer.

**Overall Objective:** Do some basic data analysis

Objective: Make new text file for the new code

Action:

1. Type nano AnalysisTest.sh
2. Copy code into file and use this for future edits

Result: Success

Objective: Separate out the tags

Action:

1. Google how to grep all lines starting with a phrase <https://unix.stackexchange.com/questions/59893/grep-lines-starting-with-1-in-ubuntu>
2. Try adding cat “$2”.txt grep ‘^tags |’ > output2.txt to current code
3. Remove the rm output2.txt section of code so I can see if it works

Error: cat: grep: No such file or directory cat: ^tags |: No such file or directory

Solution: change code to grep '^tags |' "$2".txt > output2.txt

Result: Success, output2.txt has a list of the tags used.

Objective: Have a list of the most commonly used tags

Action:

1. Google how to Analyse text data in bash
2. Find <https://towardsdatascience.com/text-mining-on-the-command-line-8ee88648476f>
3. Add cat output2.txt | tr 'A-Z' 'a-z' | tr -d [:punct:] | tr -d [:digit:] > output3.txt to my code so that I can change any capitals and remove punctuation so I can analyse it
4. Remove the rm output3.txt section of code so that I can see if it works

Error: I just realised this will leave the word tags which I don’t want to analyse. I’ll need to remove this first using sed 's/tags |//g' output2.txt > output3.txt. Including the | means that if I happen to use the word tag as a tag, it won’t delete.

1. Tolkenise words using output2.txt | tr -sc ‘a-z’ ‘\12’ > output3.txt
2. Produce list of most common tags using cat output3.txt | sort | uniq -c | sort -nr >> "$2".txt

Error: It counted the blank line as a word to list. However I found this <http://fibrevillage.com/scripting/619-ways-to-remove-empty-lines-in-a-file-on-linux> which also talks about using sed -i to edit a line without moving it to a new file. This could be useful for cleaning up my actual code later.

1. Add sed -i '/^$/d' output3.txt into the code to delete the blank line

Result: Success, at the end of my document there is a list of tags ranked from most common to least common.

Objective: Add a title so that I know what it is that is being analysed.

Action:

1. Google how to add to the end of a file
2. According to <https://stackoverflow.com/questions/23055831/add-new-line-character-at-the-end-of-file-in-bash> I can just use the echo command
3. Add echo "Tag Analysis" >> "$2".txt to my code

Error: Echo: command not found

Solution: I accidentally capitalised the E in echo. I just needed to uncapitalize it

Result: Success. The output at the end of my document is:

Tag Analysis

3 selfconsciousness

1 self

1 hegel

1 asymmetry

Objective: Try make a loop that would do the analysis I have developed for tags but on all the sections

1. Google how to loop through a for loop for different variables
2. Find <https://www.cyberciti.biz/faq/unix-linux-iterate-over-a-variable-range-of-numbers-in-bash/>The array loop seems promising
3. Try the following:

array=( 'created |' 'updated |' 'user |' 'uri |' 'text |' 'exact |' 'tags |' )

for c in "${array[@]}

do

grep '^$c' "$2".txt > output2.txt

sed 's/$c//g' output2.txt > output3.txt

cat output3.txt | tr 'A-Z' 'a-z' | tr -d [:punct:] | tr -d [:digit:] > output2.txt

cat output2.txt | tr -sc 'a-z' '\12' > output3.txt

sed -i '/^$/d' output3.txt

cat output3.txt | sort | uniq -c | sort -nr >> "$2".txt

echo "Analysis Complete"

done

Error: line 34: unexpected EOF while looking for matching `'', line 44: syntax error: unexpected end of file

Solution: I forgot to close off the “${array[@]} with the last “

Error: now it runs through fine but there is no output

Solution: I don’t think the $c is being recognised as a word. I’ll try surround it with ‘’

Error: grep: |: No such file or directory sed: -e expression #1, char 6: unterminated `s' command

Solution: if I get rid of the space between the word and the | the code works. However, I like that space being there in my final product.

If I change this section of code:

sed 's/":/ |/g;s/.//;s/ $//;s/,$//' output3.txt > output2.txt

sed '0~7 a\\' output2.txt > "$2".txt

To look like this:

sed 's/":/ |/g;s/.//;s/ $//;s/,$//' output3.txt > output2.txt

sed '0~7 a\\' output2.txt > "$2".txt

sed 's/":/|/g;s/.//;s/ $//;s/,$//' output3.txt > output2.txt

Then I can keep the space in the final product while getting rid of the space for the analysis.

Result: Success, I was able to get a list of the most common words for each section. However, the lay out isn’t very clear. I will want to put a space and title for each section. The list also contains a lot of words like “i” and “of” that aren’t very useful for my analysis. I will want to stop it from counting these words.

Objective: Put a space between each section of analysis

Action:

1. Google hot to add a line to the star of a document
2. Find <https://superuser.com/questions/246837/how-do-i-add-text-to-the-beginning-of-a-file-in-bash>
3. Use the sed -i '1s/^/task goes here\n/' todo.txt code mixed with my previous knowledge of the variables in my loop to write sed -i '1s/^/’$c’\n/' output3.txt. This should add a title to the start of the page depending on which variable is being analysed.
4. Use the previous technique I used to add a space every 7 lines, but this time only for the first line sed -i '0~1 a\\' output3.txt
5. Next I will use sed to delete the | that will be added to the title along with the word I want with sed -i ‘s/|//g’ output3.txt
6. Finally I will send it all over to the final document with cat output3.txt >> "$2".txt

Error: I initially forgot to write the ‘cat’ section and so nothing sent over to the final document

Result: Success, the output is much clearer. Each section is separated and titled with what is being analysed. I still have the problem of unimportant words being counted and I have also noticed that since my code deletes all numbers the date created section is useless. Output looks as follows:

created

3 t

updated

3 t

user

3 acctgeorgiarutherfordhypothesis

uri

3 httpsplatostanfordeduentriesselfconsciousness

text

3 and

2 to

2 this

2 the

ect.

Objective: Make the Analysis more useful

Action:

1. Make the code only show the 10 most common words by adding | head -10 to the end of cat output1.txt | sort | uniq -c | sort -nr
2. Go back to the website I was looking at for data analysis earlier <https://towardsdatascience.com/text-mining-on-the-command-line-8ee88648476f>
3. Go to link to common stopwords <https://gist.github.com/sebleier/554280>
4. Type nano stopwords.txt, copy the stop words to the document, Ctrl O to save the document
5. Google how to remove stopwords in one file from one file and find <https://stackoverflow.com/questions/17693126/removing-stop-words-in-one-file-from-another-file>
6. Add grep -F -v -f stopwords.txt output1.txt into my code

Error: the words were listed on the terminal screen and not the file itself

1. Try add a > output1.txt to the code

Error: output file also input file

1. Try change the output files around so that it all lines up

Error: lots of the words I want to see are deleted such as self and other

Solution: make my own list of words that I will add to/take away from as I need to

Error: Some words I want are still deleted

1. Google stop words some more
2. Find <https://stackoverflow.com/questions/3978626/shell-to-filter-prohibited-words-on-a-file> which mentions that if the -w isn’t in the command it will delete all words containing part of the stop word
3. Change the command to grep -F -v - w -f stopwords.txt output1.txt > output3.txt

Result: Success. The text analysis now showing words that are useful.

Objective: Create titles for the Note section of the document and the Analysis section for increased reability.

Action

1. Add echo "Analysis" >> "$2".txt and sed -i '1s/^/Notes\n/' "$2".txt to the relevant sections of code

Result: Success

**Overall result:** My code currently looks like the following:

wget https://hypothes.is/api/search?url="$1"

mv search?url\* original.txt

var=$(grep -o "created" original.txt | grep -c "created")

var2=$(($var+1))

i=1

until [ $i -gt $var2 ]

do

cat original.txt | awk -v FS='"id":' '{print $'$i'}' > output1.txt

grep -o '"created".\*"updated"' output1.txt > output2.txt

sed 's/"updated"//g' output2.txt >> output3.txt

grep -o '"updated".\*"user"' output1.txt > output2.txt

sed 's/"user"//g' output2.txt >> output3.txt

grep -o '"user".\*"uri"' output1.txt > output2.txt

sed 's/"uri"//g' output2.txt >> output3.txt

grep -o '"uri".\*"text"' output1.txt > output2.txt

sed 's/"text"//g' output2.txt >> output3.txt

grep -o '"text".\*"tags"' output1.txt > output2.txt

sed 's/"tags"//g' output2.txt >> output3.txt

grep -o '"tags".\*"group"' output1.txt > output2.txt

sed 's/"group"//g' output2.txt >> output3.txt

grep -o '"exact".\*"prefix"' output1.txt > output2.txt

sed 's/"prefix"//g' output2.txt >> output3.txt

((i++))

echo "An Annotation Complete"

done

sed 's/":/ |/g;s/.//;s/ $//;s/,$//' output3.txt > output2.txt

sed '0~7 a\\' output2.txt > "$2".txt

sed 's/":/|/g;s/.//;s/ $//;s/,$//' output3.txt > output2.txt

echo "Analysis" >> "$2".txt

declare -a arr=( "created|" "updated|" "user|" "uri|" "text|" "exact|" "tags|" )

for c in "${arr[@]}"

do

grep '^'$c'' output2.txt > output3.txt

sed 's/'$c'//g' output3.txt > output1.txt

cat output1.txt | tr 'A-Z' 'a-z' | tr -d [:punct:] | tr -d [:digit:] > output3.txt

cat output3.txt | tr -sc 'a-z' '\12' > output1.txt

sed -i '/^$/d' output1.txt

grep -i -F -v -w -f stopwords.txt output1.txt >output3.txt

cat output3.txt | sort | uniq -c | sort -nr | head -10 > output1.txt

sed -i '0~1 a\\' output1.txt

sed -i '1s/^/'$c'\n/' output1.txt

sed -i 's/|//g' output1.txt

cat output1.txt >> "$2".txt

echo "Analysis Complete"

done

sed -i '1s/^/Notes\n/' "$2".txt

rm output1.txt

rm output2.txt

rm output3.txt

mv original.txt "$2-original.txt"

echo "Task Complete"

And the output from this code looks like:

Notes

created | "2019-09-13T01:12:40.864229+00:00"

updated | "2019-09-13T01:12:40.864229+00:00"

user | "acct:georgiarutherford@hypothes.is"

uri | "https://plato.stanford.edu/entries/self-consciousness/"

text | "This is where the asymmetry of self and others comes in"

tags | ["asymmetry", "self-consciousness"]

exact | "Since knowledge of other minds is\ntypically considered to be open to sceptical doubt, and\nself-consciousness is not, such lines of reasoning are transcendental\narguments and so potentially open to general criticisms of that form\nof argument "

created | "2019-09-13T01:09:48.328890+00:00"

updated | "2019-09-13T01:09:48.328890+00:00"

user | "acct:georgiarutherford@hypothes.is"

uri | "https://plato.stanford.edu/entries/self-consciousness/"

text | "This is what I want to argue in my thesis"

tags | ["self-consciousness"]

exact | "is to argue that knowledge of other minds is a necessary\ncondition of the possibility of self-consciousness"

created | "2019-08-25T02:43:51.327033+00:00"

updated | "2019-08-30T08:46:31.519316+00:00"

user | "acct:georgiarutherford@hypothes.is"

uri | "https://plato.stanford.edu/entries/self-consciousness/"

text | "People to look at for my idea of the self and other: Fichte (1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a somewhat different perspective, Mead (1934; Aboulafia 1986)"

tags | ["Hegel", "self-consciousness", "self"]

exact | "Another, related tradition has argued that an awareness of subjects\nother than oneself is a necessary condition of self-consciousness (see\n \u00a74.4).\n Historical variations on such a view can be found in Fichte\n(1794\u20131795; Wood 2006), Hegel (1807; Pippin 2010), and, from a\nsomewhat different perspective, Mead (1934; Aboulafia 1986)."

Analysis

user

3 acctgeorgiarutherfordhypothesis

uri

3 httpsplatostanfordeduentriesselfconsciousness

text

2 self

2 in

1 wood

1 where

1 want

1 thesis

1 somewhat

1 pippin

1 perspective

1 people

exact

2 selfconsciousness

2 other

2 open

2 minds

2 knowledge

1 wood

1 view

1 variations

1 transcendentalnarguments

1 tradition

tags

3 selfconsciousness

1 self

1 hegel

1 asymmetry

**Week 9**

**Overall Objective:** I’m fairly happy with where my code is at the moment. There are definitely more adjustments that I could do, but I think there is a risk that I may keep fiddling with it forever and never let it go. At this stage it does what I set out to achieve. For now I want to turn my attention to packaging the code and fulfilling the other requirements for getting a HD in this class. Therefore my objective is to set up a github repository that satisfies the requirements of the documentation portion of the assignment.

Objective: Create a github repository with a license, a README, the code, the stopwords, the known bugs and the user acceptance tests

Action:

1. Open github
2. Click new repository
3. Name the repository ProofOfConcept
4. Add description “This is my FOAR700 Proof of Concept” (this will probably change at some point
5. Google licenses
6. Add GNU license. This seems most appropriate as I am happy for anyone to use/modify my code
7. Click initialise with a README
8. Create a file titled AUTHOR with my information
9. Create file titled ProofOfConcept.sh which contains the code
10. Update README.md with deployment instructions
11. Create a file titled stopwords.txt which contains the stop words
12. Create a file titled KnownBugs which contains the bug that
13. Create a folder called user acceptance tests containing a file of instructions and the tests themselves
14. Move the code/stopwords into a folder called code

Result: Success

Objective: Create a binder to satisfy the HD requirement of the deployability section

Action:

1. Google mybinder.org and read the instructions

Error: I don’t think I can use mybinder.org because my script is in bash. Although I was using SWAN, I was only ever using terminal.

Solution: Find something similar that is for terminal rather than jupyter notebook. After lots of googling <https://repl.it/> seems promising

1. Link to my git repository
2. Delete files I don’t need (so only ProofOfConcept.sh and stopwords.txt are left)

Error: It won’t let me delete or rename the main.sh file

Solution: For now I’m just going to write “This file is left intentionally blank”. I may change this later

1. Test whether I can share with someone else by sending the link <https://repl.it/@GeorgiaRutherfo/ProofOfConcept>
2. Change a bunch of things on the page when I am not on my account

Result: Success, this did not change the original document I sent but instead created a new document

1. Test that the code actually works by writing bash ProofOfConcept.sh <https://plato.stanford.edu/entries/self-consciousness/> test

Result: Success, I am able to get the code working on a number of different operating systems and the original link remains unchanged.

**Overall Result:** Success, I have documentation on github and a virtual environment for the code to run in. The next things I should work on are the automatic testing and putting the code on a static website. <https://github.com/GeorgiaRutherford/ProofOfConcept>

**Week 11**

**Overall Objective:** Set up automatic backups of my work to satisfy the HD requirement

Objective: Set up duplicati

Action:

1. Google duplicati
2. Download duplicati
3. Click add backup
4. Reconfigure a new backup
5. Name: ProofOfConceptBackUp
6. Webdav storage type
7. On cloudstor go to settings
8. Copy username
9. Copy cloudstor.aarnet.edu.au to server and port
10. Copy /plus/remote.php/webdav/ and add ProofOfConceptBackUp/ to path on server
11. Choose my FOAR705 folder as source data
12. Keep default schedule
13. Change to smart backup retention

Result: Success

Objective: Restore from backup

Action:

1. Create folder called test
2. On duplicati go to restore
3. Select ProofOfConceptBackUp
4. Select to restore all files
5. Select to restore to my test folder

Result: Success, restored

**Overall Result:** Success. I should check to make sure it is actually backing up according to the schedule though

**Overall Objective:** Add comments to my code to satisfy the HD requirement of narrative documentation

Action:

1. Add comments throughout my code
2. Go back and check my learning journal each time I have forgotten what a particular line of code does
3. Have someone read over it to make sure it all makes sense
4. Test the code still works with all the comments
5. Update my github repository with new code
6. Update my repl.it link with new code

**Overall Result:** Success

**Overall Objective:** Figure out automatic testing to satisfy the HD requirement of Quality Assurance

Objective: Try do an automatic test

Action:

1. Google unit tests in bash
2. Find bats git repository
3. Read installing bats section of README
4. download git repository zip

Error: since I’m working in SWAN I’ll need to find a way to download the repository there

1. drag downloaded zip to SWAN
2. google how to unzip in terminal
3. Find <https://askubuntu.com/questions/86849/how-to-unzip-a-zip-file-from-the-terminal>
4. Try command unzip bats-master.zip -d bats-master

Result: success, downloaded git repository is unzipped

1. bash install.sh test

**Overall Result:** Having fiddled around with Bats for a while I don’t think it will work for me. Bats seems to test what the output on the interface is rather than the content of the files. A lot of my tests (such as not using a real URL) seem to run through everything correctly but then the files have no content. I may be able to use this for some other tests though (like checking if a file exists after running a code <https://www.engineyard.com/blog/bats-test-command-line-tools>). I’ll see if something works better for me before coming back to this though

**Overall Objective:** Start on Pico Presentation Slides

Action:

1. Download template
2. Read through code trying to figure out what everything does
3. Change title to Proof Of Concept
4. Change author to Georgia Rutherford
5. Change email to [georgia.rutherford@students.mq.edu.au](mailto:georgia.rutherford@students.mq.edu.au)
6. Change institution to Macquarie University
7. Delete section starting with % picture to get rid of the cat on the top right

Error: This messed up the spacing

Solution, just delete the figure/figure1.png part

1. Rename buttons
2. Upload images by dragging them into overleaf
3. On slide six (now the beginning of Usage) change figure6 to Execution-example
4. On slide 7, change the section on tabular to the formatting for slide 9
5. Change “back to figure” to “back to usage”
6. On slide 8, copy code from slide 9 and paste before images there so that I have three photos total
7. Change width so the photos all fit on the page
8. Change photos to Code1, Code2, Code3

Error: it couldn’t find the images

Solution: I accidentally didn’t save the photos to the “figure” folder. Move the photos to there and it works

Error: The images are too small, it's hard to see what they are

Solution: I’ll put up an example section of code and then write a comment to where they can see the whole thing

**Overall result:** Success, I made my initial commit to github. I have not finished the slides yet, but I think I have gotten the hang of how to edit them.

**Overall Objective:** Do unit testing

Note: So I realised (thanks to <http://softwaretestingfundamentals.com/unit-testing/> ) I should be testing that components of my code and not the whole code. It would be useful to test that wget function works, since I know that doesn’t work in git-bash for some reason. It might also not work elsewhere. I also have had the thought that perhaps my work trying BATS in SWAN terminal the other didn’t work because I didn’t have the right sort of permission to install stuff on cloudstor. I’m going to try again on git-bash just to see if I can get some sort of unit testing working.

Action:

1. Open git-bash
2. Go to bats folder
3. Type bash install.sh trial
4. Go into newly created folder
5. Type nano testrun.bats
6. Use example test from <https://medium.com/@pimterry/testing-your-shell-scripts-with-bats-abfca9bdc5b9>
7. Save and exit file
8. Type bats. testrun.bats

Error: bats command not found. Honestly I have no idea how to do unit testing.

**Overall Result:** I’m just confusing myself. I’m going to go into the consultation hours tomorrow and ask for help.

**Week 12**

**Overall Objective:** Make a unit test work with help from Brian

Objective: Install BATS properly

Action:

1. $ git clone https://github.com/bats-core/bats-core.git
2. $ cd bats-core
3. $ ./install.sh /scratch/.local/

Error: I couldn’t figure out what the file location was meant to be, brian was able to find this for me with cd bats-core and echo $PATH

Result: Success

Objective:Write a test for wget command since I know that this is a bit of code that doesn’t work in git-bash

Action:

1. Use syntax from <https://github.com/bats-core/bats-core#installation>

@test "addition using bc" {

result="$(echo 2+2 | bc)”

[ "$result" -eq 4 ]

}

and <https://linuxize.com/post/bash-check-if-file-exists/>

FILE=/etc/resolv.conf

if [[ -f "$FILE" ]]; then

echo "$FILE exist"

Fi

To write

@test "that wget connects to API" {

wget https://hypothes.is/api/search?url=https://plato.stanford.edu/entries/self-consciousness/

[ -f "search?url=https:%2F%2Fplato.stanford.edu%2Fentries%2Fself-consciousness%2F" ]

}

Error: I had to get rid of sections referring to the “result” since I am just testing if the file actually exists

Error: had to get rid of the $ because that means something specific in bash

1. Name file wgettest.bats

Result: I think this test will work

Objective: Run test

Action:

1. Type chmod +x wgettest.bats

Note: According to Brian this will give shell permission to run the script, without this step I got a permission denied error

1. Type ./wgettest.bats

Error: Test failed because I had written the wrong file name

Solution: Change filename to correct filename

Result:

✓ that wget connects to API

1 test, 0 failures

**Overall Result:** Success, I have created a unit test

**Overall Objective:** Create a static website which has deployment, execution instructions, and example code to satisfy the HD requirement of documentation.

Action

1. Follow instructions on <https://pages.github.com/> to set up a website directly linked to my repository

Error: for some reason only the ReadMe.md file is the only file that appears on the website

Solution: after some googling and not really finding any answers I think the easiest solution to this is just to add my code to the ReadMe.md file

1. Copy and paste the code into ReadMe.md file

Error: all the # turned into titles

Solution: I’ll just remove the # and write in the section to see the git repository for the full code

Error: Now the code is formatted weirdly on the website. Some sections are underlined and some aren’t. I bet there are some symbols used in the code causing this

Solution: google and a website which talks about the formatting syntax on github <https://help.github.com/en/github/writing-on-github/basic-writing-and-formatting-syntax> try putting ‘’’ before and after the code

Error: ‘’’ did not work

Solution: looking at the website again I think it isn’t actually ‘’’ and is instead ``` I’ll try that now

Result: Success, the code section is formatted correctly

**Overall Result:** I now have a static website containing deployment, execution instructions, and example code and connected to my github repository

**Data Carpentry**

**The Unix Shell**

**Week 4**

**Overall Objective:** Complete up to the end of Working Files and Directories

Objective: Set up on my home computer

Action:

1. Download data-shell.zip
2. Move to desktop
3. Unzip data-shell.zip
4. Open <https://gitforwindows.org/>
5. Click download
6. Click open git bash
7. Type cd

Error: None

Result: Complete

Objective: Read through Introducing the Shell to remind me of what we went over in class

Result:

$ is a prompt to input instructions

ls is a list of the contents of the current directory

Mis-typed commands are the most common reason for errors

Useful command discussed in class was ctrl l which clears the screen

Objective: Exploring More ls Flags

Action:

1. Type ls -1

Error: I put a space between the - and the 1 to start with. Solution was to take away that space.

Error: Just realised that it is an L and not a 1

Solution: Type ls -l

1. Type ls - l -h

Result: List is clearer, command -h makes it human readable

Objective: Listing Recursively and By Time

Action:

1. Type ls -R -t

Result: Files sorted by the time of last change. Lots of text appeared on the screen and showed no sign of stopping, so I closed and reopened it to move on.

Objective: See inside desktop

Action:

1. Type ls -F Desktop

Error: ls: cannot access 'Desktop': No such file or directory

Solution: As I reopened git-bash I was in the wrong directory. I had to type cd again to get back to the home directory and then it worked.

Result: Success, the data-shell folder is in the list

Objective: Look in data-shell

Action:

1. Type ls -F Desktop/data-shell

Error: The displayed list is just data-shell/

Solution: Because of how I unzipped the file there is a folder data-shell called data-shell. To solve this type ls -F Desktop/data-shell/data-shell to see inside that second folder

Result:Success

Objective: Change directory

Action

1. cd desktop
2. cd data-shell
3. cd data-shell
4. cd data

Note: The instructions say it doesn’t print anything but this is incorrect for me. It tells me which directory I am in:

georg@DESKTOP-LFLG4GE MINGW64 ~

$ cd desktop

georg@DESKTOP-LFLG4GE MINGW64 ~/desktop

$ cd data-shell

georg@DESKTOP-LFLG4GE MINGW64 ~/desktop/data-shell

$ cd data-shell

georg@DESKTOP-LFLG4GE MINGW64 ~/desktop/data-shell/data-shell

$ cd data

georg@DESKTOP-LFLG4GE MINGW64 ~/desktop/data-shell/data-shell/data

Typing pwd does also tell me which directory I am in.

Result: Success

Objective: Move up a directory

Action:

1. Type cd . .

Error: bash: cd: too many arguments

Solution: I originally thought there was a space between the two .’s. Remove this space and it works.

Objective: Move to home directory and then back into data

Action:

1. Type cd on its own (takes you back to the home directory)
2. Type cd Desktop/data-shell/data-shell/data (move back to data in one step)
3. Check where I am with pwd and ls -F

Result: Success

Objective: Absolute vs Relative Paths

To get back to their home directory they could use option 5, 8 or 9

Error: they could also use option 7 which is cd~/data/..

Result: Almost correct, the solution said option 7 would also work but is needlessly complicated

Objective: Relative Path Resolution

The command ls -F ../backup would display the fourth option because .. would take you back up into users and then /backup would take you from users into backup. Then the list of files in backup is original/ pnas\_final/ pnas\_sub/.

Result: Correct

Objective: ls Reading Comprehension

Both option 2 and 3 would result in that output

Result: Correct

Objective: Make a directory

Action

1. Type mkdir thesis
2. Check that directory was made by typing ls -F

Result: success thesis directory is in the list

Objective: Create a text file

Action

1. Type cd thesis to move into the thesis directory
2. Type nano draft.txt to make a new file
3. Type text
4. Press ctrl O to save
5. Press ctrl X to quit out of text editor

Result: Success

Objective: Creating Files a Different Way

Action:

1. Type touch my\_file.txt
2. Type ls -l

This made a file which contains no data. I am really not sure when you would want to make a file like this.

Result: Apparently some programs require there to be an already existing empty file which they can populate. So the touch command allows you to make blank text files.

Objective: Rename file

Action:

1. Type mv thesis/draft.txt thesis/quotes.txt
2. Type ls thesis

Result: Success. Although the command mv means “move” it can be used to rename

Objective: Move quotes.txt to current directory

Action:

1. Type mv thesis/quotes.txt

Error: mv: missing destination file operand after 'thesis/quotes.txt'

Solution: I forgot to say where it was going. Type mv thesis/quotes.txt . to move to current directory

Result: Success

Objective: Moving to current folder

The answer will be mv ../analyzed/sucrose.dat ../analyzed/maltose.dat .

Result: Success

Objective: Renaming Files

The only correct option is 2.

Result: Success

Objective: Moving and Copying

The second option is correct, as the file protiens-saved.dat was copied to the directory above the one we are looking into

Result: Success

Objective: Using rm Safely

Typing rm -i thesis\_backup/quotations.txt will ask for confirmation before deleting the file. This is useful so we don’t accidentally delete important things.

Result: Success. Also interesting to note that Unix shell doesn’t have a trash bin. Also rm will not delete a directory unless we also type -r. For example rm -r thesis would delete the thesis directory.

Objective: Copy with Multiple Filenames

Action:

1. Type cp amino-acids.txt animals.txt backup/
2. Type ls -F backup
3. Type cp amino-acids.txt animals.txt morse.txt

The first set of files were copied to the backup directory. The second set of files received the error “cp: target 'morse.txt' is not a directory”.

Result: Success. The last item on a list of things to copy must be where to copy them to.

Objective: List filenames matching a pattern

Option 3 produces ethane.pdb methane.pdb.

Result: Success

Objective: More on Wildcards

$ cp \*dataset\* backup/datasets

$ cp \*calibration.txt backup/calibration

$ cp 2015-11-\* send\_to\_bob/all\_november\_files/

$ cp \*23 send\_to\_bob/all\_datasets\_created\_on\_a\_23rd/

Error: I forgot that she only wanted to send bob datasets. So the last command should be $ cp \*-23-datasets send\_to\_bob/all\_datasets\_created\_on\_a\_23rd/.

Objective: Organizing Directories and Files

Action:

1. Type cp fructose.dat sucrose.dat analyzed/
2. Type rm fructose.dat sucrose.dat

Result: I think my commands probably would have worked, but the solution was mv \*.dat analyzed. I was copying the files and then removing them afterwards instead of just moving them. Their solution would have been quicker.

Objective: Reproduce a folder structure

The first two options would work . The second option would take a bit longer as it involves actually moving between the directories.

The third option would fail as the 2016-05-20 directory has not been made yet.

In the fourth option they only changed directory into the 2016-05-20 directory before making the other two directories. Therefore they would not be inside the data directory as desired.

Result: Success

**Overall Result:** Completed up to the end of Working Files and Directories

**Week 5**

**Overall Objective:** Finish the Unix Shell

Objective: What Does sort -n Do?

Typing sort -n makes it sort numerically.

Result: Success

Objective: What Does >> Mean?

Typing >> adds the text to the existing file, whereas > overwrites the text.

Result: Success

Objective: Appending Data

Action:

1. Type head -n 3 animals.txt > animals-subset.txt

Error: head: cannot open 'animals.txt' for reading: No such file or directory

Solution: I forgot to change directories, I was still in molecules. Delete animals-subset.txt file using rm, move to data using cd ~/Desktop/data-shell/data-shell/data/.

1. Type head -n 3 animals.txt > animals-subset.txt
2. Type tail -n 2 animals.txt >> animals-subset.txt

Answer: Option 3, the first 3 and the last 2 lines of animals.txt are copied to animals-subset.txt

Result: Success

Objective: Piping Commands Together

Option 4 is correct. First you want to get the number of lines, then you want to sort it numerically so the top numbers are the smallest, then you want to see the first 3, and you use the pipe to connect them.

Result: Success

Objective: Pipe Reading Comprehension

The answer is 2012-11-05,raccoon, 2012-11-06,deer, 2012-11-06,rabbit. The first step shows what is in the list, the second step takes the first 5 options, the third step takes the last 3 from those 5, and the last step sorts the in the reverse order.

Result: Success

Objective: Pipe Construction

Type cut -d , -f 2 animals.txt | uniq

Error: I need to sort first because the uniq command just cuts out adjacent lines

Solution: Type cut -d , -f 2 animals.txt | sort | uniq

Result: Success

Objective: Which Pipe?

Option 4 is the correct answer. The first step cuts based on the comma and then shows the second column, the next step sorts, the final step shows a count of all animals with the same name next to each other.

Result: Success

Objective: Wildcard Expressions

You would need to write them out separately as \*A.txt or \*B.txt. This would show an error if there were no files with the A or B.

Result: Success

Objective: Removing Unneeded Files

The solution is rm \*.txt as this would remove all files that end with .txt.

Result: Success

Objective: Variables in Loops

The first code outputs a list of the .pdb documents each time it encounters a document ending in .pdb.

The second code lists the file it is encountering every time it encounters a .pdb file.

Result: Success

Objective: Limiting Sets of Files

1. Option for is the correct answer. Everytime the code encounters a file starting with c, it lists that file.
2. Option 4 is the correct answer. Everytime the code encounters a file containing the letter c, it lists that file.

Result: Success

Objective: Saving to a File in a Loop - Part One

Option 1 is correct. The > command was used and so the file will be overwritten each loop until the last. To avoid this the >> command should have been used.

Result: Success

Objective: Saving to a File in a Loop - Part Two

Option 3 is correct. Since the >> command was used all files ending in .pdb will be saved to the file all.pdb.

Result: Success

Objective: Doing a Dry Run

We would want to run the second option because it will echo what is in the quotations onto the screen, rather than echoing something into the file analysed-$file.

Result: Success

Objective: Nested Loops

A folder is made for each species/temperature combination.

Result: Success

Objective: List Unique Species

Action:

1. Type species.sh

Error: I forgot to write the nano and got the error bash: species.sh: command not found

Solution: Type nano species.sh

1. Type:

# Usage: bash species.sh filename

cut -d , -f 2 "$1" | sort | uniq

Error: I keep trying to click to move which point of the text I am editing, you have to use arrow keys.

Result: I missed the part where it said it needed to be able to take any number of file names. The real solution is a loop:

# Script to find unique species in csv files where species is the second data field

# This script accepts any number of file names as command line arguments

# Loop over all files

for file in $@

do

echo "Unique species in $file:"

# Extract species names

cut -d , -f 2 $file | sort | uniq

done

Objective: Why Record Commands in the History Before Running Them?

If something goes very wrong you can see what happened without actually repeating it.

Result: Success

Objective: Variables in Shell Scripts

Option 2. The first line is head -n 1 \*.pdb, this will get the first line of all files ending in .pdb. The second line is, tail -n 1 .pdb, this will get the last line of all files ending in .pdb.

Result: Success

Objective: Find the Longest File With a Given Extension

The script would look something like:

wc -l $1/\*.$2

sort -n

tail -n 2

head -n 1

Error: I keep forgetting that you can use pipelines to put things all on one line. I believe my version would still work if you were running it from a file.

Result: Success

Objective: Script Reading Comprehension

The first script will repeat any files with a . in the name.

The second script allows you to input 3 different files and it will show the contents of those files.

The last script will repeat all .pdb files.

Result: Success

Objective: Debugging Scripts

Echo command not working due to a typo of $datafile as $datfile

Result: Success

Objective: Using grep

Option 3 is correct. The command -w restricts it to whole words, which is why the word “of” is included but not “software”

Result: Success

Objective: Tracking a Species

cut -d : -f 2 | cut -d , -f 1,3 | grep -w $1 -r $2 > $1.txt

Error: I mixed up the order

Solution: grep -w $1 -r $2 | cut -d : -f 2 | cut -d , -f 1,3 > $1.txt

Note: the instructions say to use man grep, but the man command doesn’t work on git bash

Objective: Little Women

I had trouble with this answer. I think I am still a little unclear on loops and should go over the information again. The answer is:

for sis in Jo Meg Beth Amy

do

echo $sis:

grep -ow $sis LittleWomen.txt | wc -l

done

Objective: Matching and Subtracting

Option 1 is correct.

Result: Success

Objective: find Pipeline Reading Comprehension

This script counts the number of lines of all files ending with .dat and lists them numerically.

Result: Success

**Overall Result**: Complete

**Open Refine**

**Week 6**

**Overall Objective:** Finish Data Carpentry up to the end of Filtering and Sorting with OpenRefine

Objective: Set up for my home Computer

Action:

1. Download Open Refine for windows
2. Unzip
3. Run openrefine.exe
4. Download data file

Result: Success

Objective: Using Facets

The issues with the data from the village column are that there are spelling errors and differences. For example Chirdozo vs Chirodzo and Ruaca - Nhamuenda vs Ruaca-Nhamuenda.

Exercise:

1. There are 19 different dates
2. The column is formatted as text
3. The most interviews were collected in November

Result: Success

Objective: Transforming Data Exercises

Exercise 1: value.replace("’", "") , value.replace("]", "") , value.replace(" ", "")

Exercise 2: The mobile\_phone and radio are most commonly owned and the computer and car are least commonly owned

Error: Parsing error at offset 12: Missing number, string, identifier, regex, or parenthesized expression

Solution: initially I wrote value.split(;) but you need the “” so it is value.split(“;”)

Exercise 3: Farmers were most likely to lack food in November

Exercise 4: History/Reuse sped up the process of cleaning the rest so fast

Result: Success

Objective: Using undo and redo

Very useful to be able to see all the steps and to undo/redo up to a particular step

Result: Success

Objective: Filtering

1. Mabatipitched and mabatisloping (you can find this quickly by using the text facet)
2. Writing mabatip or mabatis

Result: Success

Objective: Excluding Entries

Clicking include/exclude in the text facet window changed which roof type was displayed.

Result: Success

Objective: Sort

Sorting GPS altitude by numbers smallest to largest shows that some of the GPS altitudes are listed as 0.

Result: Success (likely this information was missing and not incorrect)

Objective: Sorting by multiple columns

When sorting by longitude and latitude the village 49 was between two Chirodzo entries. When searching by date the 49 village was also in a large group of Chirodzo villages. I think village 49 is Chirodzo.

Result: Success

**Overall Result:** Complete up to the end of Filtering and Sorting with OpenRefine

**Week 7**

**Overall Objective:** Finish Open Refine data carpentry

Objective: Numbers Exercise

For all the columns with numbers, the numbers in the cell turned green and changed formatting from the left to the right. For the village column that had no numbers in it nothing changed and the message “text transform on 0 cells” appeared

Result: Success, the transform to numbers only works on numbers

Objective: Numeric Facet

I changed three cells in the no\_membrs column. I changed one to abc, one to null and one to blank. The numeric facet indicates there are two non-numeric cells and one blank cell. When I uncheck the numeric box but leave the non-numeric and blank boxes checked I can see the cells I changed.

Result: Success

Objective: Saving your own work as a script

Action:

1. Click undo/redo
2. Click extract
3. Copy and Paste the Json script to a new file
4. Open new project of the messy dataset
5. Click undo/redo
6. Click apply
7. Paste script

Result: Success. I wonder if this would have been an easier way to clean some of my hypothes.is work. I think I would have run into issues with it not being in .csv format (which it couldn’t be at the start). Plus I kind of like that I know what all my code is doing rather than just copy/pasting without understanding each step. Still, it is useful to know that I could do this if I ever have to analyse multiple spreadsheets in the future.

Objective: Exporting

After I unzipped the file it made a new folder containing two other folders, one called history and a zipped one called data. There are also lots of other options on how to export the file (such as .xls or .csv)

Result: Succuss

Objective: Using Other Resources

I looked at the grateful data github link. It ran through some similar stuff on setting up and cleaning in open refine, but what I found interesting was that it also discussed some ways you could analyse data using open refine. The example they gave was that faceting tool could be used to analyse who an artist often wrote songs with. The faceting tool would provide a list from least to most common

Result: Success

**Overall Result:** Success, completed OpenRefine data carpentry.

**Week 10**

**Overall Objective:** Finish up to Introducing dplyr and tidyr

Objective: Use the install option from the packages tab to install the ‘tidyverse’ package.

Action: Click install, type tidyverse, click install, wait

Result: success

Objective: What do you think is the current content of the object area\_acres? 123.5 or 6.175?

Action: 123.5

Error: I hadn’t rerun the line yet and so it was still 6.175

Objective: Create two variables length and width and assign them values.

Action:

length <- 5

width <- 2

area <- length \* width

area

[1] 10

If I change length/width area is still 10

length <- 3

width <- 2

Area

[1] 10

Result: Success

Objective: We’ve seen that atomic vectors can be of type character, numeric (or double), integer, and logical. But what happens if we try to mix these types in a single vector?

Action: R changes them to the same type

Result: Success

Objective: What will happen in each of these examples?

Action: R changes them to be the same type (so the TRUE changes)

Result: Success

Objective: How many values in combined\_logical are "TRUE" (as a character)

Action: There are two

Error: Apparently R doesn’t have memory of past data types. Since the TRUE in num\_logical is changed first there is only 1

Objective: Using this vector of rooms, create a new vector with the NAs removed, Use the function median() to calculate the median of the rooms vector, Use R to figure out how many households in the set use more than 2 rooms for sleeping.

Action:

1. rooms <- c(1, 2, 1, 1, NA, 3, 1, 3, 2, 1, 1, 8, 3, 1, NA, 1)
2. rooms\_no\_na <- rooms[!is.na(rooms)]
3. median(rooms, na.rm = TRUE)
4. rooms\_above\_2 <- rooms\_no\_na[rooms\_no\_na > 2]
5. length(rooms\_above\_2)

Result: Success

Objective:

1. Create a data frame (interviews\_100) containing only the data in row 100 of the interviews dataset.

interviews\_100 <- interviews[100, ]

1. Notice how nrow() gave you the number of rows in a data frame?

n\_rows <- nrow(interviews)

interviews\_last <- interviews[n\_rows, ]

1. Use nrow() to extract the row that is in the middle of the data frame. Store the content of this row in an object named interviews\_middle.

interviews\_middle <- interviews[(n\_rows / 2), ]

1. Combine nrow() with the - notation above to reproduce the behavior of head(interviews), keeping just the first through 6th rows of the interviews dataset.

interviews\_head <- interviews[-(7:n\_rows), ]

Result: Success

Objective: Rename the levels of the factor to have the first letter in uppercase: “No”,”Undetermined”, and “Yes”. Now that we have renamed the factor level to “Undetermined”, can you recreate the barplot such that “Undetermined” is last (after “Yes”)?

levels(memb\_assoc) <- c("No", "Undetermined", "Yes")

memb\_assoc <- factor(memb\_assoc, levels = c("No", "Yes", "Undetermined"))

plot(memb\_assoc)

Result: Success

Objective: Using pipes, subset the interviews data to include interviews where respondents were members of an irrigation association (memb\_assoc) and retain only the columns affect\_conflicts, liv\_count, and no\_meals.

Action:

interviews %>%

filter(memb\_assoc == "yes") %>%

select(affect\_conflicts, liv\_count, no\_meals)

Result: Success

Objective: Create a new data frame from the interviews data that meets the following criteria: contains only the village column and a new column called total\_meals containing a value that is equal to the total number of meals served in the household per day on average (no\_membrs times no\_meals). Only the rows where total\_meals is greater than 20 should be shown in the final data frame.

Action:

interviews\_total\_meals <- interviews %>%

mutate(total\_meals = no\_membrs \* no\_meals) %>%

filter(total\_meals > 20) %>%

select(village, total\_meals)

Result: Success

Objective: How many households in the survey have an average of two meals per day? Three meals per day?

Action: 52, 79

Result: Success

Objective: Use group\_by() and summarize() to find the mean, min, and max number of household members for each village.

Action: Chirodzo 2-12, God 3-15, Ruaca 2-19

Result: Success

Objective: What was the largest household interviewed in each month?

Action: 19, 12, 17, 15, 15

Result: Success

**Overall Result:** Success, completed up to the end of Introducing dplyr and tidyr

**General**

**Common Errors**

Error: get command doesn’t download what you wanted it to

Solution: wget is the command to download from the web

Error: unexpected EOF while looking for matching `"' or line 2: syntax error: unexpected end of file

Solution: Don’t copy and paste from learning journal/change the format of the “”. In fact if you ever get an error after using a script from the text file just check all the “” are correctly formatted.

Error: sed command not working as expected

Solution: Check that you are using /’s to indicate words and not “‘s like you would use with grep.

Error: Certain symbols do not count as text in LaTex

Solution: Most of the time writing \ or \text before the symbol fixed this. For the > symbol you have to write $>$.

Error: While unit testing you get permission denied

Solution: Type chmod +x (filename).bats before trying to run a bats test

Error: in github the # part of a code is making a heading (or other parts of a code are messing with formatting)

Solution: Put ``` before and after the section of code (make sure its the back tick and not the ‘)

**Common Commands**

wget <https://api.hypothes.is/api/annotations/>{AnnotationID} can get the information of one annotation

wget <https://hypothes.is/api/search?url=>{URL} can get the information of all annotations on a particular website

nano creates a document that I can write code in

mkdir makes a new file

sed -i deletes from the file you are working on without having to move it around