**Learning Journal**

**Weeks 4-6**

**Ellen Kirkpatrick**

**Week 4**

 The Unix Shell Software Carpentry

Downloaded set-up in class.

Navigating files and directories

1. Inputted ls. Successful (colours also automatic on my laptop - do not have to input code for colours).
2. Inputted pwd. Successful.
3. Inputted -F (switch or flag) . Not successful. Shell responded with command not found.
4. Re-attempted with ‘ls -F’ (adding in ls and a space). Successful. Same results as ls as colours automatically there.
5. Requested help. Successful.
6. Requested manual. Not successful.
7. Inputted ‘ls -1’. **Result**: Listed the directories (same as ls command). I cannot see the difference.
8. Inputted ‘ls -1-h’.  **Result**: Not successful. Computer came back with a message saying ambigious. Read through help and it refers to h as human readable. I can’t get it to work with -1 option. Only by itself.
9. Inputted ls -R. Successful but it removed $. Did not know how to get it back. Closed down and started again for next command.
10. Inputted ls -t
11. **Objective:** Determine what order ‘ls -R -t’ displays things in? Process - inputted command. **Result:** sorted by time of last change.
12. Inputted desktop command. Successful.
13. Inputted command to locate data-shell. Successful.
14. Followed and copied other commands in lesson. Successful.

Absolute vs relative paths acitivty - How can Amanda get back to her home directory?

**Solution:** cd by itself takes you back to the home directory.

Also cd ~ (which I didn’t know). The other 2 options for yes I was aware of but they are longer steps.

Relative path resolution activity - What will ls -F ../backup display?

**Solution:** Option 4. Did not get this right away. Had to look at solution and work out why it was correct. Consulted back to the commands. Used help command and understood why it was the right answer.

Comprehension activity - What commands will display output?

**Solution:** Option 2 - correct. Shortcut. Option 3 also correct but lists the absolute path.

Working with files and directories

1. Followed lesson. Created thesis directory. Successful.
2. Created new tex file. Successful.
3. **Error -** Was successful in creating directory and file. But they did not appear in the right place. They are not in the data-shell folder. They are on the drive but somewhere else.
4. Tried to exit with command. **Error -** Unsuccessful. Closed down and started again. How to exit?!

Creating files in a different way activity

1. Inputted command. **Result -** file did not show up. Unsuccessful?
2. Inputted second command. **Result -** there was a file created and I can see it in the list. **Error -** I cannot see the size of it. The solution says 0 bytes but I cannot see this.
3. I could not think of a reason why to create a file this way. Read solution. Would be useful for programs that already require some kind of output or file.

**Error -** Tried to follow instructions on renaming the file within the new thesis directory. My directory appeared elsewhere on the computer and I don’t know how to access it through shell. I can’t practice this part of the lesson on the required file. Need help to find the files and how to create them in the right place. Read through the instructions of this part of the lesson.

Moving files activity - What does she need to move the file away from wrong folder?

**Solution:** Use .. to return to folder before and use /xxxxx (name of the folder).

Renaming files activity - How to rename file?

**Error -** Did not know answer. Had to look at solution to understand.

Copy with multiple filenames activity -

1. Inputted first instruction. Unsuccessful. Computer responded saying ‘cannot stat...’. Don’t know why this happened. Followed instructions clearly.
2. **Error** - Need help to go through this lesson. Entering commands as shown on the online lesson but not going through as desired.

Elaboration I assignment

**Aim:** Elaborate on week 4 scoping to identify technologies that can accomplish each step.

1. Talked about week 4 scoping with group (Jeremy and John).
2. Opened up voyant online (which was mentioned in the week 4 lecture) and tested. Uploaded various sources from a semester 1 assignment.
3. **Result:** Voyant deemed useful. It is able to accomplish the first 3 steps identified in scoping exercise II.
4. Narrowed the focus of scoping II to focus only on identifying connections between different sources and exporting references with sources if they are decided to be relevant. This is a more prominent area in research.
5. Started elaboration on Latex document. Changed settings to auto-compile.
6. Applied codoing which I have learnt to format the structure.
7. Found program Zotero. Installed. Identified as a useful program for storing sources, adding metadata, linking sources through tags and exporting reference details.
8. Decided to use voyant and zotero to accomplish the tasks set out in scoping.
9. Voyant - identifies key themes for further searching and draws connections between sources. Zotero - stores the sources with additional metadata and connects the sources through tags and exports reference details. Cloudstor - additional cloud storage for extra back up/
10. Speculated as to whether shell can be used to more efficiently connect these programs?
11. **Problems:** Texts are a little difficult to read on Voyant, could be more user friendly. Despite the usefulness of all 3 programs, they are currently disconnected. Is there a way to link these? Possibly sourcetree? Or is there a cloudstor syncing function? (don’t know how to use this).

Latex

Had an easier time using Latex this time as I already knew how to itemise, this is the only formatting required for this assignment.

Was still using \\ for vertical line breaks. Imported ‘[Demonstration-of-crossreftools](https://www.overleaf.com/project/5d6726aca15546264a1624cd)’ from github. Helpful in understanding new coding commands. Applied

\setlength{\parindent}{0em} for the indent length of the paragraph. Applied

\setlength{\parskip}{1em} for whitespace between paragraphs.

% before typing will make it a comment and it won’t be included in the recompiled version.

Committed to github more often to start taking advantage of version control. Previously, would only commit the final document. This way I can track changes over time.

New programs found

Voyant - online platform. tested by putting in journal articles from a past assignment. it succesfully picked up on the key subject areas.

Zotero - installed and beginning to test (extracting metadata from browsers and adding tags).

Sourcetree - downloaded but not tested.

Attempt 2 at Software Carpentry episodes

Spoke to Brian. Problems with my journal led to difficulties in solving problems. Decided to have a second attempt at the episodes and follow the steps again. This time recording each command inputted into shell and its outcome.

Navigating files and directories

1. Find out where we are. run pwd. Result: users/ellen. Successful.
2. List contents of filing system. run ls. Result: shows list of contents in different colours. Successful.
3. Make list more comprehensible. Run ls -F. Result: no change, list already comprehensible with colours (/ directory, @ link and \* executable already present).
4. Access help. run ls — help. Result: help menu appears. Successful
5. Access manual. run man ls. Result: shell responds with command not found. Not accessible on this computer. Help is the only option.
6. List contents recursively. Run ls -R. Result: successful.
7. List contents by time of last change. run ls -t. Result: successful.
8. Get to desktop. Run ls -F Desktop. Result: listed desktop contents. successful.
9. Access data-shell folder. Run ls -F Desktop/data-shell/. Result: opened contents of data-shell folder. Successful.
10. Change locations to data directory. Run cd Desktop, cd data-shell, cd data. Result: Moves from desktop directory to data shell-directory to data directory. Successful.
11. Move out of the directory to one before. Run cd .. . Result: moved back into data-shell directory. Successful.
12. Test what cd without an argument does. Run cd. Run pwd after. Result: cd without an argument takes you back to the home directory.
13. Get back to data directory in one step. Run Desktop/data-shell/data. Result: successful. back in data directory.

Working with files and directories

Crearing a directory

1. Ran cd Desktop, cd data-shell to det to data shell directory.
2. See contents. ran ls.
3. Create new directory called thesis. ran mkdir thesis.
4. Ran ls again in data-shell. thesis directory appeared. successful.

Create a text file

1. Get to thesis directory.  Ran cd thesis. Successful - now in thesis directory.
2. Create text file. ran nano draft. Successful - opened into a text editor.
3. Wrote a few lines.
4. Save file. Pressed ctrl + O and inputted file name. Pressed enter and draft accepted.
5. Checked thesis folder manually on computer. Tex file appeared. Successful.
6. Return to shell. press ctrl X . Text editor closed and return to shell. Successful
7. Ran ls on shell to double check file. Successful.

Create file differently

1. Create file. ran touch my\_file.txt.
2. Inspect files. ran ls -l. File appeared. Successful.
3. Check size. Looked at number next to date. The file size is 0 bytes as it is empty.

Moving files and directories

1. Return to data-shell directory. Run cd ~/Desktop/data-shell/. Successful. In data-shell directory.
2. Change file name. ran mv thesis/draft.txt thesis/quotes.txt. File name changed to quotes. Successful.
3. Move file to data-shell directory. ran mv thesis/quotes.txt . (note the dot preceded by s space). Successful.
4. Check file is no longer in thesis directory. Ran ls thesis. File no longer appears there.

Copying files and directories

1. Copy file from data-shell to thesis directory. ran cp quotes.txt thesis/quotations.txt.
2. Ran ls quotes.txt thesis/quotations.txt. Successful. Copy of file now in thesis directory.
3. Copy directory using recursive option. Ran cp -r thesis thesis\_backup.
4. List contents to check original thesis and new thesis\_backup contents. Ran ls thesis thesis\_backup. Successful. 2 directories with contents.

Removing files and directories

1. Return to data-shell directory. Ran cd, pwd, cd Desktop, cd data-shell. Result - returned to data-shell directory.
2. Remove the quotes.txt file. Run rm quotes.txt.
3. Confirm file has been removed. Run ls quotes.txt. Result - computer responds with no such file or directory. Successful.
4. Execute rm -i thesis\_backup/quotations.txt. Result: will prompt for removal. Type y or n to confirm deletion. Typed y. Result, file removed. Successful.
5. Remove thesis directory. Run rm thesis. Result - unsuccessful. Rm only works on files, not directories.
6. Run rm with recursive option to remove directory. Run rm -r thesis. Successful - thesis directory removed. Tested with ls thesis.

Operations with multiple files and directories

1. Get to data directory. ran cd data. Successful, in data directory.
2. Create backups for multiple files at once. Ran mkdir backup. Then cp amino-acids.txt animals.txt backup/.  Result: backup folder now in data with 2 files.

**Week 5**

Could not attend lecture, had to listen online. Second half of lecture was a bit difficult to listen to, it went in and out of being understandable.

Unix Shell software carpentry

Pipes and filters

1. Get to molecules directory in data-shell folder. Ran cd ~/Desktop/data-shell/molecules, ls. Result - in molecules directory with files listed.
2. Ran command wc \*.pdb to get a word count of files in directory. Successful. Files displayed with word count.
3. Ran wc -1 \*.pdb. Result - computer responded with unknown option.
4. Which file contains fewest lines. Ran wc -l \*.pdb > lengths.txt. Result - computer responded with unkown option.
5. **Error -**Unsure why these commands are not being recognised, played around with spacing. But tested ls lengths.txt in next step and the txt file appeared. Not sure what happened.
6. Test file exists. Ran ls lengths.txt. Result - lengths.txt exists (checked manually also, it does exist but not sure how if I received an error message from previous step with computer).
7. Output page by page. Ran less lengths.txt. Displays screenful of file. Pressed q to exit.
8. **Exercise -** Run sort “sort -n” on a file, the output is numbers appear in numeric order. This is because -n specified a numerical order rather than alphanumerical.
9. Sort files numerical order. Ran sort -n lengths.txt. Computer had no response with files. Just went to the next $.
10. Proceeded with: $ sort -n lengths.txt > sorted-lengths.txt and $ head -n 1 sorted-lengths.txt . Result - the desired output was not received. Computer just remained the same.
11. **Error -** There are text files for lengths and sorted lengths but they do not contain text as they are meant to, nor did shell show the output as expected in the software carpentry lessons.
12. Exercise - what does >> mean? Ran echo hello >testfile01.txt and echo hello >> testfile02.txt. No visible difference but 2 files appeared and both say hello. Ran again, did not execute twice as instructions specified in first time. executed instructions twice second time. testfile01 adds to file not replaces like the first test.
13. Appending data exercise - did not understand the question. Had to look at solution to work it out for explanation. **Error -** had a spelling error. Re-attemped and understood the answer as it appeared.
14. **Error** - do not know how to input vertical bar into shell. Google search and found answer. It is on the same key as the forward slash.
15. Run sort and head together. sort -n lengths.txt | head -n 1. methane.pdb file shows.

Pipes and filters attempt 2

Due to multiple errors, decided to attempt this episode again. Just have to be aware of the new files/contents in molecules directory due to exercises in other episodes.

1. Open molecules directory. ran cd ~/Desktop/data-shell/molecules. In molecules directly.
2. List contents. ran ls . Contents shown.
3. Ran wc \*.pdb. Get word count for each pdb file. Successful.
4. Ran wc -1 \*.pdb.  **Error-** computer responds with -1 is an unknown option. (I don’t think -1 is an option on this computer and going to do this episode without using that command as it has been the cause of all errors).
5. Ran ls lengths.txt. File shown.
6. Use cat command. ran cat lengths.txt. Files appear to show contents.
7. What does sort -n do exercise - this command lists them in numeric order.
8. Ran sort -n lengths.txt. Files appear in numeric order. Successful.
9. Ran sort -n lengths.txt > sorted-lengths.txt.
10. Ran head -n 1 sorted-lengths.txt. Methane.pdb file appeared. successful.
11. What does >> mean exercise - using >> means the text in the file gets added each time rather than replacing the text that is there.
12. Appending data exercise - the command would result in the first 3 lines and the last 2 lines from animals.txt into animals-subset.txt.
13. Ran sort -n lengths.txt | head -n 1. Methane.pdb appeared. Successful.
14. Instead of running wc -1 \*.pdb | sort -n, tried wc \*.pdb | sort -n. Successful! Files apparead in numeric order as shown in exercise. -1 command is the problem and will no longer be used.
15. Ran wc \*.pdb | sort -n | head -n 1. methane.pdb file appeared. successful.
16. Piping commands together exercise - wasn’t sure on answer. Looked at solution to work out how they got there. Tried to run wc \*.pdb | sort -n | head -n 3. **Error -** had spelled head incorrectly. Tried again. Successful! Files appeared.
17. Pipe reading comprehension exercise - head extracts first 5 lines from animals.txt, tail extracts last 3 lines and then sort -2 sorts lines in reverse order and output is redirected to a new file final.txt.
18. Pipe construction exercise - looked at solution. not familiar with these commands.
19. Which pipe exercise - option 4. ran: cut -d, f 2 animals.txt | sort | uniq -c | wc - 1 to test in data directory. Manually checked. Successful.
20. Removing uneeded files exercise - want to delete processed data files and only keep raw files. would run rm \*.txt. successful.

Loops

Loops - programming construct which allow us to repeat a command/set of commands for each item in a list.

1. Get to creatures directory. Ran cd ~/Desktop/data-shell/creatures. Successful.
2. Ran head -n 5 basilisk.dat minotaur.dat unicorn.dat. Successful. Details appeared on shell.
3. Print out classification for each species given on second line of each file. Ran: for filename in basilisk.dat minotaur.dat unicorn.dat (enter) do (enter) head -n 2 $filename | tail -n 1 (enter) done (enter). Successfull. Shell showed 3 classifications.
4. Variables in loops exercise - run for datafile in \*.pdb (enter) do (enter) ls \*.pdb (enter) done (enter). Output was all files repeated. Same output on each iteration. Successful.
5. Ran for datafile in \*.pdb (enter) do (enter) ls $datafile (enter) done (enter). each file listed as a singular. Successful.
6. Limiting sets of files exercise - ran for filename in c\* (enter) do (enter) ls $filename (enter) done (enter). Only cubane.pdb listed. Successful.
7. Ran for filename in \*c\* (enter) do (enter) ls $filename (enter) done. cubane.pdb and octane.pdb listed. Successful.
8. Saving to a file in a loop (pt 1) exercise - ran for alkanes in \*.pdb (enter) do (enter) echo $alkanes (enter) cat $alkanes > alkanes.pdb (enter) done (enter). Text from each file is written to alkanes.pdb but overwirtten with each loop so the text from propane.pdb is the final content.
9. saving to a file in a loop (pt 2) exercise - ran for datafile in \*.pdb (enter) do (enter) cat $datafile >> all.pdb (enter) done. All of the text from the files will be saved to a file called all.pdb.
10. Ran for filename in \*.dat (enter) do (enter) echo $filename (enter) head -n 100 $filename | tail -n 20 (enter) done. Expands to list of files that will process. Successful.
11. Modify all files in creatures directory but save a version of the original files. Use loop: for filename in \*.dat (enter) do (enter) cp $filename original-$filename (enter) done. Checked creatures folder and there were copies made and the original files were specified.
12. Doing a dry run exercise - the difference between the two commands is the inclusion of: “ ”. Shell will show what is in quotation marks. Tested.
13. Nested loops exercise - was not sure of answer, looked at solution to work down the lines of commands to see how they got there. Ran in molecules directory: for species in cubane ethane methane (enter) do (enter) for temperature in 25 30 37 40 (enter) do (enter) mkdir $species-$temperature (enter) done (enter) done. Result: new directory created for each molecule with each number specified.

Shell scripts

1. Ran ctrl l to clear screen. Still in molecules directory.
2. Create new file. ran nano middle.sh. Opens text editor nano.
3. Inputted head -n 15 octane.pdb | tail -n 5 on nano. Colours changed as text was entered. Save file ctrl O and exit ctrl X.
4. Manually checked molecules directory. File exists. Successful.
5. Ask shell to execute the commands the file contains. Ran bash middle.sh. Output shows what the previous command would have done if run directly. Successful.
6. Open middle.sh file. Ran nano middle.sh. Text editor opens in shell. Successful.
7. Replace octane.pdb with “$1” (first filename on the command line). Run bash middle.sh octane.pdb. Successful. Output appears.
8. Run bash middle.sh pentane.pdb. Successful. Output appears.
9. Open text editor. Run nano middle.sh. Text editor opens with middle.sh file.
10. Exit ctrl X. Run bash middle.sh pentane.pdb 15 5. Successful. Output appears.
11. Change script behaviour. Run bash middle.sh pentane.pdb 20 5. Output changes.
12. Open nano with nano middle.sh. Add comments with #. Exit ctrl X.
13. Process many files in a single pipeline, sort .pdb files by length. Ran wc -1 \*.pdb | sort -n. **Error**- computer does not recognise command.
14. Open new file on text editor. ran nano sorted.sh. Successful. New file.
15. Inputted #Sort files by their length, #Usage: bash sorted.sh one\_or\_more\_filenames, wc -1 “$@” | sort -n. Saved and exited text editor ctrl O ctrl X.
16. Ran bash sorted.sh \*.pdb../creatures/\*.dat. **Error**- computer responds with unknown option -1 for wordcount.
17. Save commands in a file. Run history | tail -n 5 > redo-figure-3.sh. New file created.
18. Why record commands in the history exercise - in case a command goes wrong or to solve problems.
19. Variables in shell script exercise - created new file, ran nano script.sh. inputted heal -n $2 $1, tail -n $3 $1. saved ctrl O exited ctrl X. Ran bash script.sh ‘\*.pdb’ 1 1. The first and last line of each file displayed from molecules directory for .pdb files. Successful.
20. Find the longest file with given extension exercise - ran nano longest.sh. **Error**  - did not know what to input or whether it was necessary. Ran command bash longest.sh /tmp/data pdb as provided in exercise but computer responded with no such file or directory.

Received following output: bash: warning: shell level (1000) too high, resetting to 1

bash: warning: shell level (1000) too high, resetting to 1. Computer got very slow for a time and then started working again —> I don’t know what this is?!?! Decided to stop on shell for the evening as end of shell scripts lesson was reached.

Finding Things

Re-started following day due to the warning last night. No problems with computer or shell now.

1. Get to writing directory. Ran cd, cd Desktop/data-shell/writing, cat haiku.txt. Displays text of file on shell. Successful.
2. Find lines in file that contain the word not. Ran grep not haiku.txt. 3 lines which contain word appear.
3. Search same file for the pattern ‘the’. an grep The haiku.txt. 2 lines with word appear.
4. Restrict to word ‘the’ on its own. Ran grep -w The haiku.txt. 1 line appears. Successful.
5. Repeat step but for a phrase. Ran grep -w “is not” haiku.txt. 1 line appeared. Successful.
6. Apply -n option. Ran grep -n “it” haiku.txt. Listed number of line in file next to output.
7. Combine option with -w to find word ‘the’ and -n to number lines. ran grep -n -w “the” haiku.txt. 2 lines appeared with numbers next to them.
8. Use -i option to make case-sensitive. Ran grep -n -w -i “the” haiku.txt. Successful.
9. Use -v option to invert search. Ran grep -n -w -v “the” haiku.txt. Shows lines with numbers which don’t have word “the”.
10. Ran grep —help. Had a scroll through help menu to see what can be combined with grep. Didn’t understand how they could be used.
11. Using grep exercise - and the presence of absence would result from command: grep -w “of” haiku.txt because it looks for whole word.
12. Wildcards - ran grep -E ‘^.o’ haiku.txt. Shows 3 lines of output. Successful
13. Tracking a species exercise - man grep is not an option available on computer and grep help didn’t specify exactly how to grep text recursively. Looked at the animals.txt which is an example file. Looked at solution to work out how they got answer. Did not understand.
14. Little women exercise - looked at solution and used it to understand the question and process. I understand how they used the for and echo commands, and applying the grep. I would not have been able to develop that command by myself though.
15. Listing vs. finding - ran wc -1 $(find . -name ‘\*.txt’). Result - computer responded saying -1 was an unkown command.
16. Took out -1 option and ran: wc $(find . -name ‘\*.txt’). Successful. Received output
17. **Error** - started to look at previous errors and realised most have been linked to commands where -1 was used. Perhaps not recognised by computer and that’s why the commands were not being received.
18. Matching and subtracting exercise - find files in data directory which end in .txt but do not contain the word net. Command: find data -name ‘\*s.txt’ | grep -v net. Successful. Found this answer on my own.
19. Find pipeline reading comprehension - explain: wc -1 $(find . -name ‘\*.dat’) | sort -n. I think this script will find all files which have .dat in the name, and will list the files by their number. Successful!!
20. Finding files with different properties exercise - command “man find” does not work on this computer. Looked at solution to work out how they got there.

Elaboration II

1. Started with defining the initial key phrase to begin research. Chose “Australian welfare system” - important to learn more about the welfare system and its history and development before pursuing a certain direction in research.
2. Typed this phrase into the database and there was thousands of sources. Refined depending on year and source-type. Chose 2 peer-reviewed journal articles and 1 book chapter to test for elaboration II. These are the types of sources I will be using most in research. They will be used for analysis and interpretation of the welfare system.
3. The sources all had the appropriate metadata required to make a reference list and they could all be accessed completely online. This test decided that the Macquarie University library database can be used for further research.
4. Moved onto the second test.
5. Opened Voyant on a web browser and uploaded 10 sources which I had used from a semester 1 assignment. I am familiar with these sources and have identified common themes between them.
6. Tested Voyant with these sources to see if it could identify the themes I had used these sources for. This is to test the reliability of Voyant and to determine whether it can be used to identify themes on unknown sources.
7. Voyant successfully identified 5 key areas related to these sources and the assignment. They were: welfare, social, Indigenous, Australian and health. These sources had been initially used to find information on health outcomes of Indigenous Australians and their relationship with the welfare system.
8. Voyant passed this test. Decided Voyant is reliable and can be used to identify themes of unknown sources in order to determine relevance.
9. Test 3. Copied urls of online sources from test 1 into Voyant. **Error -** it did not successfully transfer the sources, it picked up on the information from the Macquarie University gateway site. Not successful. (This may be a problem of restricted access - sources may be able to be accessed online but they cannot be copied etc.).
10. Decided to change the order of decomposition and test zotero first and then test whether sources can be uploaded to Voyant through Zotero.
11. Test 4. Opened zotero and returned to the sources where they were accessed originally from the university database. Clicked on the Zotero icon on the web browser and added 3 sources to my library.
12. Tested on Zotero whether the metadata had transferred and could be exported. Successful.
13. Tested whether additional annotations could be made to each source, and whether tags could be created where sources could be linked together. Successful.
14. Tested whether sources could be searched by their details and tags. Successful.
15. This test found that Zotero is useful for storing sources, linking them together, exporting metadata and adding annotations.
16. Test 5. Upload sources to Voyant through Zotero. Files cannot successfully be uploaded from Zotero as the same problem happens as in test 3.
17. Returned to the original sources from unviersity database and downloaded them in .pdf format in a ‘test’ folder on the Desktop. Re-uploaded them to Voyant as pdf files to determine whether Voyant is useful in identifying commonalities between them.
18. Voyant very helpful in comparing the 3 sources and showing whether they were connected or not. The trends tool is very helpful in linking to specific paragraphs across the sources. Zotero can also connect to the contexts tool and sources can be extracted directly from there.
19. Voyant successful in picking up common themes between texts, linking specific paragraphs and for determining whether they will be helpful for research.

Summary of elaboration II: testing and revision —> to be included in assignment.

The tests in elaboration II successfully responded to the steps articulated in elaboration I. Test 3 was the only test to be unsuccessful in that sources could not be copied over to Voyant. This may be due to the fact these items have restricted access and were accessible through institutional access but not in general. The same sources could be used when downloaded first and uploaded in pdf format.

This creates an extra step in the process and is not so straightforward as outlined in the decomposition but Voyant was successful in test 5 in identifying common themes and being used to determine relevance. Zotero can also extract the sources and metadata straight from Voyant.

Elaboration II indicates that sources need to be downloaded first from the database, uploaded to Voyant and then extracted to Zotero where they can be linked through tags and additional annotations can be added. In response to the initial scoping problem, these programs are helpful in connecting sources and storing them with metadata.

Latex and Github

Latex was straightforward to use in this case as only itemise was required.

Committed to github multiple times for version control.

**Latex codes ongoing**

|  |  |
| --- | --- |
| **Code** | **Function** |
| \section | section heading |
| \\ | vertical line break (manual) |
| \textbf | bolden text |
| \begin{itemize}  \item  \end{itemize] | making bullet point list |
| \setlength{\parindent}{0em} | sets the indent length of a paragraph |
| \setlength{\parskip}{1em} | sets the whitespace between paragraphs |

**Shell ongoing**

|  |  |
| --- | --- |
| **Symbol/command** | **Function/meaning** |
| $ | Shell waiting for input |
| ctrl l | clears |
| ls | list contents of current directory |
| pwd | print working directory (where we are) |
| ls —help | request help |
| / | If at the front - it refers to the root directory  If inside a name - it is a separator. |
| ls -R | lists content of directories recursively (lists sub-directories etc.) |
| ls -t | lists things according to time of last change |
| cd | change directory (without an argument will return you to home directory) |
| * or  - - | switches/flags will begin with these.  changes behaviour of command |
| .. | the directory containing this one |
| mkdir | make directory |
| mv | move |
| rm  -i | remove (permanent as no trash bin)  will prompt for removal |
| \* | wildcard - matches zero or other characters |
| | | pipe - tells the shell we want to use the output of the command on the left as the input to to the command on the right. |
| for | repeat a command/group of commands once for each item in a list  needs a variable to refer to the thing it is currently operating on |
| $ or  > | from shell: a prompt, shell expects you to type something  from you: instruction from you that the shell should redirect output or get the value of a variable |
| !!  !$ | Retrieves immediately preceding command (like the up arrow).  retrieves the last word of the last command. |
| history | display recent commands  !number to repeat command by number. |
| # | starts a comment and runs to end of the line.  computers ignore comments. |