Module 2 – Open Notebook, Fail-Log

Exercise 1: The Dream Case

- This Exercise taught me to use well managed databases to search for desired research data
  - The process was simple and easily accomplished

- Databases explored
  - Epigraphic Database Heidelberg (http://edh-www.adw.uni-heidelberg.de/inschrift/suche)
    - Search word "figlina"
    - Studied "Result 1" (http://edh-www.adw.uni-heidelberg.de/edh/inschrift/HD002817)
      - Created a nano file "Module 2 – Exercise 1 - EpigraphicDatabaseHeidelberg.md"
      - Copied data to file
      - Committed the file
        - $ git add -A
      - Created .html copy (then uploaded to hist3814o GitHub repo)
        - $ pandoc -o Module2-Exercise1-EpigraphicDatabaseHeidelberg.html Module2-Exercise1-EpigraphicDatabaseHeidelberg.md
  - Commonwealth War Graves Commission, Find War Dead (https://www.cwgc.org/find/find-war-dead)
    - Searched last name "Jooste": https://www.cwgc.org/find/find-war-dead/results?lastName=Jooste
      - Downloaded Microsoft Excel file then uploaded to hist3814o repo
      - Created a nano file "Module 2- Exercise 1 - CommonwealthWarGravesCommission,FindWarDead.md"
      - Committed the file
        - $ git add -A
      - Created .html copy (then uploaded to hist3814o GitHub repo)
        - $ pandoc -o Module2-Exercise1-CommonwealthWarGravesCommission,FindWarDead.html Module2-Exercise1-CommonwealthWarGravesCommission,FindWarDead.md

Exercise 2: Wget

- The instructions for this Exercise were easy to follow and all went smoothly
  - The process did take a very long time. However, that was simply due to lengthy wget download durations

    I had to redo part of this Exercise due to an error message received during Exercise 6
    - An explanation is provided at that point
- Performed Ian Milligan's wget tutorial at the Programming Historina (https://programminghistorian.org/en/lessons/automated-downloading-with-wget#step-two-learning-about-the-structure-of-wget--downloading-a-specific-set-of-files)
  - Made new DH Box directory "wget-activehistory"

        - $ mkdir wget-activehisotry
    - Entered new directory
        - $ cd wget-activehistory
    - Downloaded index page of http://activehistory.ca/papers/
        - $ wget http://activehistory.ca/papers/
    - Learned various "Option" commands for wget
        - -r → Recursive retrieval to a depth of five sites after the first
        - --no-parent → Wget stop following links past parent directory
        - -l 2 → Wget only follows one link after initial
        - -l 3 → Wget only follows two links after initial
        - -w 10 → adds a ten second wait in between server requests
        - --random-wait → varies the wait by 0.5 and 1.5 times the value you provide
        - --limit-rate=20k → limits bandwidth max. download speed to 20kb/s
        - -m → Mirrors/backs up an entire website
    - Downloaded all the ActiveHistory.ca papers
        - $ wget -r –no-parent -w 2 –limit-rate=20k http://activehistory.ca/papers/
            - Trailing slash critical, or wget will think it's a file rather than a directory!
    - Checked history then piped list into a new file "Module 2 – Exercise 2 - tut1commands.md"
        - $ history
        - $ history > Module2-Exercise2-tut1commands.md
        - Converted to an .html file (then uploaded to hist3814o GitHub repo)
            - $ pandoc -o Module2-Exercise2-tut1commands.html Module2-Exercise2-tut1commands.md
- Performed Kellen Kurschinki's wget tutorial at the Programming Historian (https://programminghistorian.org/en/lessons/applied-archival-downloading-with-wget#recursive-retrieval-and-sequential-urls-the-library-and-archives-canada-example)
    - Note, much of the following commands were provided by the course Worksheet, and are not mentioned in the tutorial…
    - This needed to be done responsibly and respectfully so as to not appear like a bot attacking the site
    - Made a new directory: "war-diary", entered new directory, then ensured I was in the parent
        - $ mkdir war-diary
        - $ cd war-diary
        - $ cd ~
    - Used a Python script to gather all URLs concerning the diary images from the 14th Canadian General Hospital war diaries, as documented by the Library of Archives Canada (http://collectionscanada.gc.ca/pam_archives/index.php?fuseaction=genitem.displayItem&lang=eng&rec_nbr=2005110&rec_nbr_list=3366167,3203123,2005097,2005100,2005101,2005099,2005096,2005110,2005108,200510)
        - Python file created: "urls"
            - $ nano urls.py
        - Added text to grab 80 URLs from e001518029 to e001518110
            Script can be found under point #4 at http://workbook.craftingdigitalhistory.ca/module-2/Exercises/#exercise-2-wget
        - Ran the Python script
            - $ python urls.py

- Examined the file, then exited
    - $ nano urls.txt
- All the URLs downloaded from the urls.txt file with wget
    - $ wget -i urls.txt -r --no-parent -nd -w 2 --limit-rate=100k
    - I should note here that I encountered an error message in Exercise 6 in which I attempted to convert the first file to TIFF with ImageMagick
        - The message was informing me that the file I was looking for did not exist in "war-diary"
        - The reason for this is that I accidentally entered the wrong directory with the "$ cd ~" command
        - This downloaded all the files into the master branch parent directory instead or "war-diary"
        - In order to correct this mistake, I redid everything after the creation and entering of the "war-diary" directory
            - This time ensuring is was actually in said directory!!
                - $ pwd
- **Note: This bullet point was performed prior to realizing my mistake noted above**
    - Checked history, then piped list into a new file "Module 2 – Exercise 2 – tut2commands.md"
        - $ history
        - $ history > Module2-Exercise2-tut2commands-rectifiedversion.md
        - Converted to an .html file (then uploaded to hist3814o GitHub repo)
            - $ pandoc -o Module2-Exercise2-tut2commands.html Module2-Exercise2-tut2commands.md
- Due to the failure mentioned, this point was redone as well
    - New file name "Module 2 – Exercise 2 - tut2commands - rectifiedversion.html"

Exercise 3: TEI

- In this exercise, I did some basic marking up of a text using standards from the Text Encoding Initiative (http://www.tei-c.org/index.xml)
    - I was somewhat confused at one point when I couldn't find the .xsl file, but the problem was soon rectified
- Downloaded, as a Zip file, the "module3-wranglingdata" repository (https://github.com/craftingdigitalhistory/module3-wranglingdata)
    - Opened the "tei-hist3907" file
- Vetting the Website "Recovered Histories" (http://www.recoveredhistories.org/)
    - A brief explanation as to why I consider the site to be a trustworthy provider of historical texts can be found in my blog (http://dannyjooste.ca/)
- Finding a Source
    - Browsed collection with title name *Negro Slavery* and found the pamphlet written by Zachary Macaulay
- Transcribing the page

    - Transcribed page 75 from Macaulay's *Negro Slavery* into "blanktemplate.txt", opened in Sublime Text
        - Transcribed file renamed and saved as "Module 2 – Exercise 3 - Page_75_of_ Macauley's_Negro-Slavery_ transcribed.xml"
    - Encoded Dr. Williamson's name twice and the word "he" when referring to the same man
        - <persName key="Last, First" **from**="YYYY" to="YYYY" role="Occupation" **ref**="http://www.website.com/webpage.html"> </persName>
    - Encoded Europe and Jamaica, as well as "that devoted country" when referring to Jamaica
        - <placeName key="Sheffield, United Kingdom" **ref**="http://tools.wmflabs.org/geohack/geohack.php?pagename=Sheffield&params=53_23_01_N_1_28_01_W_type:city_region:GB"> </placeName>
    - Encoded various claims and arguments
        - <interp key="reason" n="citation" cert="high" **ref**="http://www.website.com/webpage.html"> </interp>
- Uploaded the "Module 2 – Exercise 3 - Page_75_of_ Macauley's_Negro-Slavery_ transcribed.xml" to the hist3814o GitHub repo
- Was initially confused as to what the .xsl file referred to was
    - Eventually found it after reading on into "Viewing Encoded Text", seeing "000style.xsl" and remembering there was a file named that exact same thing inside the module3-wranglingdata-master folder
    - Renamed the file to "Module 2 – Exercise 3 - 000style" for personal categorization and simplicity sake, then uploaded it to the hist3814o GitHub repo
- Attempted to view the .xml file in Firefox
    - Received a XML Parsing Error → mismatched tag. Expected: </p>.
    - I believed this to be incorrect, as I did encode the starst and ends of the paragraphs correctly
    - I sent a Zulip message, along with screenshots to Dr. Graham, who was equally at a loss
    - I genuinely have no idea how or why, but I retried the process two days later, without making any changes to the file itself, and it worked perfectly.
        - I'm very confused as to how or why this happened.

Exercise 4: APIs

- This Exercise was very easy to understand and was accomplished without a hitch
- Entered the Canadian Discovery Panel (http://search.canadiana.ca/)
    - Searched "Ottawa" with the time period parameters ranging from 1800-1900
        - The API is the resulting URL (http://search.canadiana.ca/search?q=ottawa&field=&df=1800&dt=1900)
            - Includes the search words/dates
    - Formatted the data in a way that makes sense to a machine
        - Added "&fmt=json" to the url
- Learned definitons for:.
        - curl → program for downloading webpages
        - jq → deals with JSON
        - sed and awk → search within and cleaning up text
- Installed jq program in DH Box

    - $ sudo apt-get install jq -y
- Made new directory: “m2e4”, then entered it
    - $ mkdir m2e4
    - $ cd m2e4
- Made an empty file for the program, in a shell script, then opened it
    - $ touch canadiana.sh
    - $ nano canadiana.sh
- Ian Milligan’s script to retrieve oocihm (https://ianmilligan.ca/api-example-sh/)
    - Changed the script so that it points to the API at http://search.canadiana.ca/
    - Copied, and saved, script into canadiana.sh file
        - Script can be found in Exercise 4, point #8, at http://workbook.craftingdigitalhistory.ca/module-2/Exercises/#exercise-4-apis
- Told DH Box that it is alright to run the program
    - $ chmod 755 canadiana.sh
        - “chmod” command means change mode
- Ran the program → remember the “./”!
    - $ ./canadiana.sh
- Created .html file “Module 2 – Exercise 4 – commands”, uploaded to hist3814o repo
    - $ pandoc -o Module2-Exercise4-commands.html Module2-Exercise4-commands.md
- Downloaded “output.txt” file to computer from File Manager > m2e4 folder

Exercise 5: Mining Twitter

- I encountered an error code, which is illustrated at that point in the sequence
    - I was forced to halt this Exercise, and skip to Exercise 6, until solution could be ascertained
- Created a Twitter account under the username “BubblesMcGee”
- Created a Twitter application called “BubblesMcGee-twarc”
    - Used the Crafting Digital History website for the url: http://site.craftingdigitalhistory.ca/
- Copied the consumer key and the consumer secret to a TXT file
- Generated an access token and an access secret, then saved them in the TXT file
- Downloaded an older version of Twarc to work with the DH Box
    - Before typing out the full code as illustrated in Workbook, I remembered seeing a Zulip message made by Lauren Rollit, in which she encountered an error message following her own download
        - Dr. Graham looked into the matter and suggested she try removing “sudo”
        - New command entered:
            - $ pip install https://github.com/DocNow/twarc/archive/v1.2.0.tar.gz
        - However, I then encountered a different error message than Lauren, and sent replied to the Zulip conversation with a screenshot
        - Dr. Graham then prompted me to use a command to make the apt database refresh itself
            - “$ sudo apt-get update”
        - I Googled the error message : “Command python setup.py egg_info failed with error code 1 in /tmp/pip-iLBCrK-build”

        - I clicked on the first link presented, which was a group conversation discussing way to counter the issue when it was found by another researcher on a different project
            - https://github.com/facebook/prophet/issues/418
        - I took he advice by one individual to use a command to upgrade my setup tools first
            - $ pip install --upgrade setuptools
            - However, I was required to add "sudo" when DH Box informed me that I did not have permission to perform the desired task.
        - After this, the twarc download seemed to respond favourably and I was prompted to enter my app's information
- Entered my consumer secret etc., after typing:
    - $ twarc configure
- Tried searching for a small group of Tweets relating to JSON regarding hist3814
    - $ twarc search hist3814o > search.json
- Installed a command that can convert the JSON to CSV table format
    - $ sudo npm install json2csv --save -g
- Converted JSON to CSV
    - json2csv -i search.json -o out.csv
- Checked history and created .html file ("Module 2 – Exercise 5 – commands") to upload to hist3814 repo
    - $ history
    - $ history > Module2-Exercise5-commands.md
    - $ pandoc -o Module2-Exercise5-commands.html Module2-Exercise5-commands.md

Exercise 6: Using Tesseract to turn an image into text

- Like Exercise 5, I encountered another error message which prevented me from continuing until I had a viable solution for fear of making the situation worse
    - The issue is presented later, at that point in the sequence
- Converting images in the Command Line
    - Made a new directory "ocr-test", then entered it
        - $ mkdir ocr-test
        - $ cd ocr-test
    - Installed Tesseract in Dh Box
        - $ sudo apt-get install tesseract-ocr
        - Typed "Y" when prompted, to allow the use of 78.4 MB of disk space
    - Installed ImageMagick
        - $ sudo apt-get install imagemagick
    - Attempted to convert the first file to TIFF with ImageMagick
        - $ convert -density 300 ~/war-diary/e001518087.jpg -depth 8 -strip -background white -alpha off e001518087.tiff
        - Received an error message stating "unable to open image ... No such file or directory"
        - Posted a message over Zulip explaining my issue
        - After I eventually realized the mistake I made back in Exercise 2, this process worked fine
    - Extracted the text

        - $ tesseract e001518087.tiff output.txt
    - Downloaded the resulting output.txt file through the File Manager
- Converting images in R
    - Opened R Studio in Dh Box
    - Opened new blank script
        - Green "+" > R Script
    - Pasted script provided in Exercise 6 – Converting images in R, point #3: http://workbook.craftingdigitalhistory.ca/module-2/Exercises/#exercise-6-using-tesseract-to-turn-an-image-into-text
        - Saved the file as "ocr"
        - Process installed Magick, Magrittr, and Tesseract to R Studio
    - Opened Command Line
        - Note, I was given an error message here, as did a few other classmates it would seem
            - Dr. Graham recommended we first use the command "$ `sudo apt-get update`"
    - Installed the libcurl library
        - $ sudo apt-get install libcurl4-gnutls-dev
        - Typed "Y" when prompted, to allow the use of more disk space
    - Installed the libmagick library
        - $ sudo apt-get install libmagick++-dev
        - Typed "Y" when prompted, to allow the use of more disk space
    - Installed the libtesseract library
        - $ sudo apt-get install libtesseract-dev
    - Installed the libleptonic library
        - $ sudo apt-get install libleptonica-dev
    - Installed the English Tesseract library
        - $ sudo apt-get install tesseract-ocr-eng
    - Installed the poppler cpp library
        - $ sudo apt-get install libpoppler-cpp-dev
        - Typed "Y" when prompted, to allow the use of more disk space
    - Opened R Studio to run 4 "install.packages" lines
        - "magick", "magrittr", "pdf tools", and "tesseract"
    - Loaded the libraries
        - Ran each "library()" line
    - Ran each line up to "image_ocr()"
    - Exported the OCR to a text file
        - Ran the last line: "write.table()"
    - Downloaded both "output.txt" and "R.txt" files from File Manager
        - The OCR created in the command line seems much easier to read and interpret than the one created within R
- Progressively converting our files with Tesseract
    - Took screenshots of the text files: "output_1.png" and "R_1.png"
    - Uploaded both files into DH Box via the File Manager
    - In Command Line
        - $ tesseract output_1.png output_1.txt
    - Changed script's file path In R Studio
        - Find under point "Progressively converting our files with Tesseract, #4, at http://workbook.craftingdigitalhistory.ca/module-2/Exercises/#exercise-6-using-tesseract-to-turn-an-image-into-text
    - Loaded the libraries again and re-ran each line, minus the "install.packages()" lines

    - Downloaded “ouput_1.txt” and “R_1.txt” from File Manager
        - Both files now appear easier to read when compared to the originals
    - Created a new directory “`war-diary-text`”
        - Used the R method because it is faster
            - First line
                - Changed “ocr-test” in the file path to “war-diary-text”
                - Changed “e001518087” to “e001518029”
                    - Continued up to “e001518040”
            - Ran all lines minus the “install.packages” lines
- Batch converting image files
    - Replaced the old R Studio script with new
        - Found under “Batch converting image files” at http://workbook.craftingdigitalhistory.ca/module-2/Exercises/#exercise-2-wget
        - Attempted to OCR all the Canadian war diary .jpg files located in the “war-diary” directory into .png and .txt files
            - The .txt files would contain a suffix “-ocr.txt”
        - However, nothing seems to happen when I run any of the lines
            - There is not error message, simply nothing happens at all
            - Tried giving it time, and left it alone for an hour, but no change
            - Sent out a group Zulip message to ask any classmates if they were experiencing the same
            - Shay Ishola found the solution
                - She recommended we add a “~” prior to the “/” in the path
                - This seems to have rectified the issue