**Learning Journal weeks 6-8**

**Ellen Kirkpatrick**

**Week 6**

Open Refine data carpentry lessons

Working with openrefine

1. start openrefine. double click on .exe. file. Got taken to a blank black screen with lots of output getting written. It took some time but then openrefine came up in internet explorer.
2. Clicked create project, get data from, this computer. Selected SAFI messy file (had downloaded from data carpentry website). Uploaded data. Clicked create project.
3. Use faceting to look for potential errors. Scroll to village column, click down arrow and chose facet, text facet. Left panel a box appeared with all unique values in village column.
4. Sorted by name and count. data is problematic as is not consistent and some have spelling errors. Some have text names, others have number names.
5. Hovered mouse over names in facet list. Edit function becomes available.
6. Edited the errors so entries are all consistent. The name 49 cannot be made consistent with other data.
7. Applied text facet to interview date column. There are 19 options available. Column is formatted as text.
8. Produce timeline display for interview\_date. Select edit cells, common transfers, to date. Column converted to dates. Most dates were collected in November 2016.
9. Clustering - in village facet in left panel, click cluster. In pop up, selected key collision method and metaphone3 keying function. 2 clusters identified.
10. Click merge box beside each cluster and click merge selected and recluster.
11. Changed spelling of names, more clusters identified.
12. **Note:** clicked on the merge more than once, solutions for later exercises may be different.
13. Transforming data. Click items owned column, edit cells, transform.... New window opens. Can type general refine expression language (GREL).
14. Remove all left square brackets. In expression box type value.replace(”[”, “”), click ok. No longer left square brackets in item\_owned column.
15. Use same strategy to remove single quote marks, right square brackets in items\_owned column. Used the following GREL as directed in solution: value.replace("'", ""), value.replace("]", ""), value.replace(" ", ""). All on separate lines. Mostly successful, but can still see occasional right bracket next to a value. Otherwise, all replaced with semi-colons.
16. Ran value.replace("]", "") for a second time on its own. Successful. All right brackets disappeared.
17. Use text facet to see what items were commonly owned. Click items\_owned, choose facet, custom text facet. Type value.split(”;”) in expression box. Click ok. Box appears on left panel with all items owned.
18. Which 2 items are most commonly owned? Sorted by count. Mobile phone and cow plough.
19. Which 2 are least commonly owned? Solar torch and solar panel.
20. Used same steps, cleaned the months\_lack\_food column. Got error at last custom text facet.**Error-** Left panel appeared saying: *Parsing error at offset 12: Missing number, string, identifier, regex, or parenthesized expression.* How to fix???? Not sure where I went wrong or how to recover steps.
21. Copied and pasted the commands from solution earlier to get exact typing right for the months\_no\_water column. Successful.
22. Added GREL language. Text facet appeared on left.
23. Repeated cleaning and GREL steps for liv\_owned, res\_change, no\_food\_mtiigation columns. All successful.
24. Repeated GREL command using history tab. Clicked reuse next to command.
25. Using undo and redo. Click undo/redo on the left. All operations/commands are listed.
26. Clicked on a number of past commands to see the change in the dataset. Tried to see where I want wrong with the error in months\_lack\_food column but could not work it out.
27. Trim leading and trailing whitespace. Create new text facet for respondent\_wall\_type. Edited data so that spelling was correct and condensed identical choices.
28. Remove whitespace. Choose edit cells, common transforms, trim leading and trailing whitespace. Only 4 choices in text facet in left panel.

Filtering and sorting with openrefine

1. Click on arrow next to respondent\_roof\_type, select text filter. Facet appears in left.
2. Type mabat, press enter. The column shows the rows with matching info. There are 58.
3. Change view to show 50 rows. Will see most of matching rows.
4. What roof types are selected? Mabatisloping and mabatipitched.
5. How would you restrict this to only one type? Be more specific with choices of letters/word in filter.
6. Excluding entries. Create text facet for respondent\_roof\_type. Drop down menu facet, text facet. Facet appears in left panel with 2 entries that agree with text filter.
7. Use include/exlude to select entries from one of these roof types. Hovered mouse over the entry and clicked include on mabatisloping. Other entry is excluded.
8. Sort. Select drop down arrow in gps\_altitude column. Select sort by numbers and smallest first. Column sorted in order. First few values are 0. This may be problematic, maybe missing data?
9. Clicked sort again, and selected reverse to reverse sort. Successful.
10. Sorting by multiple columns. Sort on gps\_longitude with number largest first. Successful.
11. Sort gps\_lattitude as number with largest first. Successful.
12. Select drop down arrow on village column. Select edit column, move column to end. Column moves to end so it can be compared easier with GPS coordinates.
13. Find village 49. Can’t see what village it is based on coordinates.
14. Sort interview\_date column by data. Select sort, date.
15. Move village column to start. Drop down, edit column, move to beginning.
16. Row for 49 corresponds with interview times for Chirodzo village. Unsure about GPS coordinates.
17. Open text facet for village. Drop down arrow, facet, text facet. In left panel, select edit on 49. Change 49 to Chirodzo to condense entries.

**Proof of Concept - ongoing, week 6 & 7**

Made a new project on github. But at first it was in ellenkirkpatrick, not the FOAR705 team. Had to create a new repositry within FOAR705 and then make a project within that. Not sure how really to use the project management tool at this stage.

Making notes in a word document at the moment for possible user stories. Not sure how many are needed, unsure how categories work.

Will transfer these notes onto overleaf at a later stage once ideas are more developed. Once on overleaf will copy over any additions to github for version control.

—>

Started proof of concept on overleaf. Structure: listed user stories under category headings and acceptance critera. Committed to github. Created a new repositry and within that, started a new project. Deleted previous repositry so that everything (overleaf commits and project) in the same repositry.

Added in user stories to github project. Did not add acceptance criteria, not sure if meant to.

User stories are:

**Identify commonalities:** As a research student, I would like a program which can compare multiple sources at once to identify common themes and terms. This will help determine relevance of source to project.

**Extract Information:** As a research student, I would like a storage program that can extract source and metadata from an online platform so I can create references and citations more efficiently.

**Store sources:** As a research student, I would like a program that can store multiple sources with metadata so I can access all research material in the same place.

**Annotations:** As a research student, I would like to be able to add annotations to stored sources so they can be referred to more easily in the future.

**Grouping:** As a research student, I would like to link specific sources through tags or labels so they can be grouped together.

**Export metadata:** As a research student, I would like to export metadata to a word processing program, such as Microsoft Word, in order to automate the reference list.

Acceptance criteria:

Identify commonalities:

* Find peer-reviewed journal article sources on multiple databases.
* Download sources to computer.
* Upload sources to Voyant.
* Use the trends and cirrus tools on Voyant to identify key themes and terms.
* Use reader tool to read abstracts and determine relevance.

Extract information:

* Open Zotero application on computer.
* Use contexts tool on Voyant to connect to Zotero.
* Extract source and metadata from Voyant onto Zotero for storage.

Store sources:

* Check that the extracted source appears in Zotero.
* Click on each source to ensure metadata is available.
* Click on each source to make sure the original website or pdf can be accessed.
* Flag source if metadata or part of metadata is missing, or if the source is not available.

Annotations:

* Add annotations to sources in Zotero.
* Access annotations in future, and add to them more.

Grouping:

* Add tags or labels to source according to relevance or specific theme.
* Search for specific groups of sources by these tags.

Export metadata:

* Export source metadata from Zotero to Microsoft Word.
* Check that metadata is correct.
* Add to bibliography list if required.

Pre-requisites:

* All user stories must be completed in order for full efficiency.
* Voyant must have been tested prior with known sources to determine whether this is reliable. (completed in elaboration)
* Zotero must be installed on computer and a connecting icon added to the browser of choice (Mozilla Firefox) before beginning. (completed in elaboration)

Quality assurance

I don’t understand this component of PoC, or how to link it to the user stories. Is this a process like elaboration - testing the functionality??

Started to use the project management on github, but can’t see how to do this within the project tab. It seems like just columns where you can move tasks. Might just list out tests in learning journal?

Did add an initial issue identified in elaboration, that urls cannot be copied to Voyant. It works better with uploading pdf sources. I added this in the issue tab in PoC repositry, and will continue to add issues. But this tab is not linked with the project tab.

Proof of concept - update 2

Got some feedback from Osmond after I offered a few comments on his proof of concept design. He recommended to specify what user stories were essential for the PoC and which were just extra. —> took on feedback and created a new section “Notes on User Stories” in overleaf draft.

Essential user stories - identifying commonalities and storing sources. These user stories link to the central aim of being able to compare multiple sources at once and storing them in a single place for more efficient use.

Extra user stories - extracting information and exporting metadata are also crucial to the PoC in terms of functionality. But they are there to make the entire process more efficient. The annotations and grouping are not essential overall but help make the process more impressive as they are helpful for researchers referring to work in future.

After completing this section, committed to github.

Quality Assurance

Still not sure exactly what I need to do in this section but specified what the tool/program needs to do to be successful. They are the following:

Identify commonalities - For Voyant to be successful, it must identify commonalities between the different sources. At least 3 key terms will be identified and will be reflected in the trends tool. It must also show where in the source is each term used.

Extract Information - For Zotero to be successful, it must be able to extract both the source and the metadata from Voyant in one process. There must not be two separate processes for extracting the source and the metadata.

Store Sources - For Zotero to be successful, the source and metadata should be stored within the program library. They should be able to be accessed at future points and the original source (whether it is a website, or a file) can be accessed.

Annotation - For Zotero to be successful, annotations should be able to be added to specific sources. These annotations should be saved and can be edited, or updated at future points of time.

Grouping - For Zotero to be successful, tags should be able to be added to sources depending on relevance, topic area or the needs of the researcher. A search of these tags in Zotero should produce all sources under this tag.

Majority focus on zotero, but as the identifying commonalities user story is identified as crucial, it means Voyant is required to be successful.

Commited to github after writing this section.

Created a second project in github, just for quality assurance tests. Don’t have time to complete these tests before Friday. But can enter them in a to-do list. Made blank project. The user story project is in Kanban, and I am having difficulties modifying these settings and don’t know how to use cards. So thought blank may be more applicable. Put all tests into to-do column.

Still can’t see how to export a report, but I downloaded a tex. file from the repositry and added to my proof of concept submission folder on cloudstor.

Committed updated journal to github.

**Week 7**

Openrefine data carpentry

Examining numbers in openrefine

1. Opened openrefine program.
2. Selected open project from lefthand side panel. Clicked on existing project from week 6. Same data set. SAFI\_openrefine\_messy.
3. Made sure text filter facets are removed from left panel. None showing. No windows opened.
4. Transform cells in years\_farm column to numbers. Click down arrow on column > edit cells > common transforms > to number. Numbers changed to colour green and went to the right of the cell. Successful.
5. Repeat the steps from step 4 for columns: no\_members, yrs\_liv and buildings\_in\_compound. Numbers turned green and went to the right for all 3 columns.
6. Repeat steps for village column. Data entries are not numeric, they do not change and the following note appears: **Text transform on 0 cells in column village: value.toNumber().** Data cannot be changed.
7. Numeric facet exercise - edited years\_farm, first 2 rows with abc & second row changed to blank. Used pulldown menu on column and selected facet > numeric facet. Facet appears in left side panel.
8. Facet shows a graph with multiple check boxes, some checked, others not. Can see in column how many entries are numeric, text, blank or errors.
9. Remove facet by clicking the cross in top corner. Does not undo changes to data.
10. Clicked on undo/redo in top left panel. Went back to 2 steps previous. Checked data in years\_farm went back to original data. Successful.

Using scripts

1. In undo/redo section, click extract. Opens new panel where you can see operation history and select steps you want to apply by clicking check boxes.
2. Copy code from right hand side and pasted into new notepad file. Saved as .txt file in FOAR705 week 7 folder on computer. Named: openrefine data carpentry.
3. Apply script to another dataset. Open new project.
4. Load in same messy dataset. Changed the title to project 2.
5. Click undo/redo in lefthand panel. Click apply and paste contents from .txt file.
6. Click perform operations. Checked data, it has updated to reflect changes made to previous dataset. Same formatting and operations were there.

Exporting and saving data from openrefine

1. Stayed in project 2 from previous episode.
2. Export a project. Click export button in top right. Select Export Project. A pop up appears asking to save to local or google drive (this is different to what the episode says - clicked local).
3. New tab appears with tar.gz file (as outlined in episode). Selected download and file went to downloads folder. Also selected save as, and saved to FOAR705 folder.
4. Download successful. Shows zip file. Looks like it contains all of the history of operations and the data (assuming raw data).
5. Import existing project: click open in top right > import project > select the tar.gz file. Successful. Data appears in openrefine.
6. Exporting cleaned data. Click export in top right. Select comma-separated values (csv) format. Saved to downloads and to FOAR705 folder.

Other resources in openrefine

1. Read through the website links provided.
2. Opened openrefine documentation wiki site. It has features, “recipes” and examples which could be useful reference tools.

Reflection on openrefine

Out of the data carpentry lessons so far, I found this the most helpful and interesting as I could see how it could be applied in future study/work, or be relevant to me. Although I am still not overly confident in my openrefine skills, it is something I will consider using in the future as it offers more flexibility than excel and I felt I had more control over what I was commanding.

My problem was that I couldn’t work out exactly where my error back in the working with openrefine episode was, despite using the undo/redo functions to try and see where it was.

Quality assurance testing for proof of concept - github projects

1. All quality assurance tests in to do column in github project.
2. Moved testing Voyant into in-progress.
3. Used same test as done in elaboration to test usefulness of voyant. Uploaded 10 well known sources to see if it can identify common themes. Successful.
4. Uploaded 4 new sources which are unread and unknown. They have been downloaded for research on another subject’s major essay assignment. I am looking for sources that are looking at kicking away the ladder of development for developing countries, and how advanced countries have carved themselves a path in the international order.
5. Voyant shows the following key terms: countries, trade, economic, world, developing. I can see where these terms are in each journal and how often they are used. I can also see that other terms such as tariff, cotton, protection and industry are common. This is important as they are specific areas that need to be studied when answering this question.
6. Happy with Voyant’s comparison of these articles.
7. Entered these comments into github project for this card.
8. Moved on to second assurance test. Seeing if zotero can extract the metadata and the source direct from Voyant.
9. Zotero has a connector icon in Firefox browser. Also right clicking on the documents within Voyant, can save directly to Zotero library. Sources appear in Zotero library with their available metadata.
10. Tested storing sources. Closed down Zotero and re-opened in ten minutes to see if sources are saved.
11. Sources still appeared in library and could be accessed. Successful. Added notes on test in card and moved to complete column in github (this has been done for all tests).
12. Testing annotation. Selected a source, looked at info tab in top right corner. This contains all metadata and bibliographic details. Clicked on notes tab.
13. There are currently 0 notes. Clicked ‘add note’. Could add a note to the source with its relevance or anything I thought was required. This note appears connected to the file in the library. There is also opportunity to add multiple notes, or to remove notes. These notes appear in the program library under the original source.
14. Adding annotations through notes is successful. Moved to complete column.
15. Moved the grouping card to in progress column. Need to test whether tags can be added to sources in zotero and whether they can be grouped together by these tags.
16. Selected a source. Clicked on tags tab in top right corner.
17. Some tags already appear which Zotero have picked up on. I didn’t know about this and was useful to see what has been identified. Clicked on add tag.
18. Can add a one-line tag for each source.
19. Repeated the same tag for another source.
20. Performed a search for this tag through the searchbar. The two sources appeared.
21. Test successful. Tags can be added and sources can be grouped together, searched and identified by these tags. Moved to complete column.
22. All quality assurance tests for programs have been completed and tested. This particular project complete on github. Had no distinct issues in the quality assurance tests. Did not add issues in repositry tab.
23. **Problem:** When moving the cards from in progress to to-do columns in github project, they became out of order. I had them in order of user stories but they are now mixed up in complete column. Still also can’t see how to extract a report from github projects to put in cloudstor or download. How to do this??

User stories on github project

1. Opened second project. Testing user stories to see if they can be completed. Followed same pattern as quality assurance tests.
2. Moved user story identifying commonalities to in progress column.
3. This can be completed through Voyant. Voyant was successful in quality assurance, therefore can be used. Outlined test completed and moved to complete column.
4. Moved extracting information column to in progress.
5. Provided instruction on how source information and metadata can be extracted directly from Voyant to Zotero.
6. Started storing sources, moved to in progress.
7. Provided instruction on where sources are stored, where they go and how they can be accessed. Moved to complete.
8. Moved annotations to in progress.
9. Provided instruction on how notes/annotations can be added to specific sources in Zotero and where they can appear. Moved to complete.
10. Moved grouping to in progress.
11. Provided instruction on how tags can be added to specific sources, how they can be linked together and identified by their tags. Moved to complete.
12. Final user story test. Moved export metadata to in progress.
13. Opened up microsoft word. There is a new zotero tab at the top of the page. Clicked it, and selected add/edit citation. Can draw direct citations and bibliography entries from Zotero.
14. Can also right click on source in Zotero library and select create bibliography from item. Selected copy to clipboard and directly pasted the entry into micrsosoft word. Can select bibliography style, selected Harvard. Can repeat same step for citation
15. Moved to complete.
16. Project complete on github.
17. **Problem:** same issue occured in quality assurance project. The cards became out of order in the complete column.

R Studio introduction

Watched a youtube introduction tutorial: https://www.youtube.com/watch?v=mcYcjH-1giM

Before meeting with Brian later this week to get some understanding of R studio prior to the meeting to ask questions. Video not that helpful, I think they’re using an older version of R.

Searched data carpentry lessons instead and started on episodes.

Project Management of rstudio

1. Create new project. Select file > new project > new directory > (empty project did not appear) > selected R project instead. Named file r-geospatial. Clicked create project.
2. Entered getwd() into r console. Responded with the user directory of the file.
3. In lower lefthand panel, created 3 new folders: data, doc and results.
4. Saved 3 datasets from online: nordic-data.csv, nordic-data-2.csv, gapminder\_data.csv into new data directory. Found this folder using the response from R studio, went to documents, found the geospatial directory and then the new data directory within that. 3 csv files are now there.
5. Downloaded the zip folder of data from data carpentry. Move downloaded zip file to data directory. Unzip all files.
6. Files all appear in data directory.

Data structures

1. Download and read file: nordic-data.csv. Save as an object named nordic. Inputted nordic < - read.csv(”data/noridc-data.csv”). Ran in console. Recieved error message.
2. Opened new file nordic.data.csv. New tab appears in top left corner, but don’t know where to input the message.
3. Tried to move on, but I can’t type in the console, the mouse won’t click.
4. Made console full page. Then it worked. Typed in nordic <- read.csv(”data/nordic-data.csv”) . Successful. Output appeared in top right corner.
5. Entered nordic$country. Returned with output of the countries.
6. Entered nordic$lifeExp. Output appeared with 3 numbers.
7. Entered nordic$lifeExp + 2. Output added 2 to each of the numbers
8. Entered nordic$lifeExp + nordic$country. Received error message and NA NA NA. Command not meaningful in R. Not meaningul as combines life expectancy and country type.
9. Entered class(nordic$lifeExp). Response was [1] “numeric”.
10. Entered class(3.14) . Response numeric.
11. Entered class(1L). Response integer
12. Entered class(1+1i). Response complex
13. Entered class(TRUE). Response logical
14. Entered class (’banana’). Response character.
15. Entered class(factor(’banana’)). Response factor.
16. How to understand this: R interprets a specific data class.
17. Load file: open file > selected nordic-data-2.csv.
18. Entered: nordic\_2 <- read.csv("data/nordic-data-2.csv") and class(nordic\_2$lifeExp). Response was factor.
19. Entered nordic\_2$lifeExp + 2. Received NA NA NA and error message.
20. Entered class(nordic). Responded with data frame.
21. Start vectors and type coercion.
22. Entered my\_vector <- vector(length = 3), my\_vector . Received FALSE FALSE FALSE.
23. Entered another\_vector <- vector(mode = 'character', length = 3) and another\_vector. Received [1] "" "" "".
24. Entered str(another\_vector). Received chr [1:3] "" "" "".
25. Entered str(nordic$lifeExp). Received num [1:3] 77.2 80 79.
26. Discussion 1 - Why is R so opinionated about what we put in our columns of data? R is trying to keep the data clean and consistent so that each column has one type.
27. Entered combine\_vector <- c(2, 6, 3), combine\_vector. Received [1] 2 6 3.
28. Entered quiz\_vector <- c(2, 6, '3'). Did not receive direct output response but a new loutput appeared in the top right corner under values.
29. Entered coercion\_vector <- c('a', TRUE), coercion\_vector. Received [1] "a" "TRUE".
30. Entered another\_coercion\_vector <- c(0, TRUE), another\_coercion\_vector. Received [1] 0 1.
31. Entered character\_vector\_example <- c('0', '2', '4'), character\_vector\_example. Received [1] "0" "2" "4".
32. Entered character\_coerced\_to\_numeric <- as.numeric(character\_vector\_example)
33. character\_coerced\_to\_numeric. Received [1] 0 2 4.
34. Entered numeric\_coerced\_to\_logical <- as.logical(character\_coerced\_to\_numeric)
35. numeric\_coerced\_to\_logical. Received FALSE TRUE TRUE.
36. Challenge 1 - why is lifeExp different between nordic and nordic 2.
37. Entered str(nordic\_2$lifeExp). Received Factor w/ 3 levels "77.2","79.0 or 83",..: 1 3 2. Data is stored as factors.
38. Ran str(nordic$lifeExp). Received num [1:3] 77.2 80 79. Stored as numbers,
39. Using combine function. Entered ab\_vector <- c('a', 'b'), ab\_vector. Received [1] "a" "b"
40. Entered combine\_example <- c(ab\_vector, 'DC'), combine\_example. Received [1] "a" "b" "DC".
41. Make series of numbers. Entered my\_series <- 1:10. Output appeared in top right panel. Not in console.
42. **Error:** Realised read the instructions wrong. Re-entered my\_series <- 1:10, my\_series. Received output in console [1] 1 2 3 4 5 6 7 8 9 10. Successful.
43. Entered seq(10). Same output received.
44. Entered seq(1,10, by = 0.1). Received 6 rows of output with multiple numbers in each row.
45. Ask questions about vectors. Entered sequence\_example <- seq(10), head(sequence\_example,n = 2). Received [1] 1 2.
46. Entered tail(sequence\_example, n = 4). Received [1] 7 8 9 10.
47. Entered length(sequence\_example). Received [1] 10.
48. Entered class(sequence\_example). Received [1] "integer".
49. Give names to elements in vector. Ran my\_example <- 5:8, names(my\_example) <- c("a", "b", "c", "d"), my\_example. Received a b c d 5 6 7 8
50. Challenge 2 - make vector with numbers 1-26. Multiply vector by 2 and give resulting names A to Z. Used solution on data carpentry to work through it. Would not have got there otherwise.
51. Entered x <- 1:26, x <- x \* 2, names(x) <- LETTERS. No response in consolse but output in top right corner. New row of data.
52. Factors.
53. Entered str(nordic$lifeExp). Received num [1:3] 77.2 80 79.
54. Entered str(nordic$year). Received int [1:3] 2002 2002 2002.
55. Entered str(nordic$country). Received Factor w/ 3 levels "Denmark","Norway",..: 1 3 2.
56. Make vector labelling nordic countries for all in study. Entered nordic\_countries <- ('Norway', 'Finland', 'Denmark', 'Iceland', 'Sweden'), nordic\_countries
57. Received [1] "Norway" "Finland" "Denmark" "Iceland" "Sweden" .
58. Entered str(nordic\_countries). Received chr [1:5] "Norway" "Finland" "Denmark" "Iceland" "Sweden".
59. Turn vector into factor. Entered categories <- factor(nordic\_countries), class(categories). Received [1] "factor".
60. Entered str(categories). Recieved Factor w/ 5 levels "Denmark","Finland",..: 4 2 1 3 5.
61. Entered class(nordic\_countries). Received [1] "character".
62. Entered class(categories). Received [1] "factor".
63. Challenge - why are these numbers used to represent these countries? Alphabetical order.
64. Challenge - is there a factor in nordic data frame? Did not understand the questino or what to do. Used solution to copy over the responses and see how they got there. Entered nordic <- read.csv(file = "data/nordic-data.csv", stringsAsFactors = FALSE), str(nordic$country). Received chr [1:3] "Denmark" "Sweden" "Norway".
65. Used colclasses solution. Entered nordic <- read.csv(file="data/nordic-data.csv", colClasses=c(NA, NA, "character")), str(nordic$country). Received Factor w/ 3 levels "Denmark","Norway",..: 1 3 2.
66. Know where baseline levels are. Entered mydata <- c("case", "control", "control", "case"), factor\_ordering\_example <- factor(mydata, levels = c("control", "case")), str(factor\_ordering\_example). Received Factor w/ 2 levels "control","case": 2 1 1 2.
67. Lists.
68. Entered list\_example <- list(1, "a", TRUE, c(2, 6, 7)) list\_example. Received successful output, the data all listed by numbers between brackets.
69. Entered another\_list <- list(title = "Numbers", numbers = 1:10, data = TRUE ), another\_list. Received right output again.
70. Entered str(nordic). Received 'data.frame': 3 obs. of 3 variables: $ country: Factor w/ 3 levels "Denmark","Norway",..: 1 3 2 $ year : int 2002 2002 2002 $ lifeExp: num 77.2 80 79.
71. Entered str(another\_list). Received List of 3 $ title : chr "Numbers" $ numbers: int [1:10] 1 2 3 4 5 6 7 8 9 10 $ data : logi TRUE.
72. Entered nordic$country. Received [1] Denmark Sweden Norway Levels: Denmark Norway Sweden
73. Entered nordic[, 1]. Received [1] Denmark Sweden Norway Levels: Denmark Norway Sweden.
74. Entered class(nordic[, 1]). Received [1] "factor".
75. Entered str(nordic[, 1]) Received Factor w/ 3 levels "Denmark","Norway",..: 1 3 2
76. Entered nordic[1, ]. Recieved country year lifeExp 1 Denmark 2002 77.2
77. Entered class(nordic[1, ]). Received [1] "data.frame"
78. Entered str(nordic[1, ]). Received 'data.frame': 1 obs. of 3 variables: $ country: Factor w/ 3 levels "Denmark","Norway",..: 1 $ year : int 2002 $ lifeExp: num 77.2

Proof of concept

Had a meeting with Brian today, was helpful. Can understand the github projects a bit more.

Need to change all the cards (my user stories) into issues, and then in issues I can add specific notes for the tests and what to do. As issues are completed, mark them as done.

Downloaded voyant server, instead of using voyant tools online. This will make it easier to transfer files around the computer, rather than having to upload and download them all the time. Hoping Voyant and Zotero may be able to connect even more efficiently.

Had to install latest Java. Removed previous java update.

Had problems with server. Would not launch!!! Used help page: https://github.com/sgsinclair/VoyantServer/wiki/FAQ---Troubleshooting

Changed the memory settings from 1024 to 512 in text-settings file as directed. Successful!!!

Voyant server now downloaded on computer and launches in browser. Will be using Voyant server from now on instead of voyant tools online.

Tested by uploading 4 files being used for a current assignment. Identified acceptable commonalities and themes shared between the text. Successful!

Have realised though that zotero cannot extract the items directly from Voyant. Might change the order of user stories.

Proof of concept order: Download sources straight to zotero library -> transfer to voyant server (hopefully via shell command) -> then manually tag and annotate data in zotero.

Looked for add ons but there are none available, or they are no longer working between zotero and voyant. Otherwise there will only be the manual way and there can be automation in this process unless shell can work. I don’t know how to do the shell commands for this.

Github projects

1. Had 2 projects running, closed one and focused all on user stories.
2. Moved all cards back to the to-do column.
3. Converted cards to issues.
4. Created a succinct title for each card.
5. Checked that all user stories now appear as issues. Successful.

Github projects update

1. Closed issue/user story: identifying commonalities. Voyant server is a useful tool for this. Can do this.
2. Started on issue/user story: extract source info from voyant.
3. Having problems opening voyant server, have changed the memory to less but still the browser won’t open. Very tempermental. May go back to using voyant tools online as this server is proving unreliable. Re-opened user story: identifying commonalities. Not going to use server, return to website.
4. Downloaded pdf journal articles directly from MQ library database to Zotero. Sources will remain in Zotero from beginning to end of process. They will also be downloaded in a new directory called “test” on the desktop. This will ensure they can be uploaded quickly to Voyant online, rather than having to upload single sources at a time.
5. Began documenting workflow process on overleaf. Committed changes regularly to github but had to create a new repositry, could not link to existing PoC repositry. This has been labelled “implementation and instructions”. The workflow process outlines the start to finish of this process, and necessary set up steps including installation of programs, updates etc.
6. This will slightly change the process, as it is no longer required for Zotero to extract information from Voyant as the sources are already stored there. Rather, Voyant will provide the necessary insight on how to annotate and tag sources for future use.