Data Mining Historical Research

CSM6720 - mah60 - Matthew Howard

May 2018

Contents

1 Literature Review

The assignment that was given to us was to be able to apply data mining techniques to historical material. This is a major issue for most historical researchers as finding methods to gather specific items of information from the data can be moved done through database management systems (DBMS) which is faster than doing the data mining manually. However, one of the problems that occur within using a database management system to do this is that to make this method as efficient as possible some form of standardisation will need to be applied to the data. This is a problem as altering the data to make the DBMS efficient and flexible could reduce the richness and detail within the data inside the databases. So a balance between these two ideas must be maintained to produce useful data for processing.

The problem that historical researchers have is rarely seen amongst other subjects, such as geography or physics. This is because their data is primarily fixed to certain values and has little error, making the data easier to read. However, with each historical research project the data gathered is sparse and has many irregularities. This is due to there being no standards to the records gathered when they were created. Further to this another problem is that there are rarely similarities found between different research projects. This means if a database and method of standardisation is developed for one set of research, this can rarely be implemented into another persons research methods.

The assigned research project that was given to us, was to look at Aberystwyth Shipping Records. This data contains information about the ships that visited Aberystwyth. Each record contained information about each ship, these were; the vessels name, official number, port of registry and mariners. The mariners also had additional data on each of the sailors, these included their name, year of birth, when they joined the ship and more information to help identify who they were and what they were doing on the ship. The project was to filter the data that was gathered to an extent where we can maintain its original richness and create a database that will allow for efficient and flexible data mining techniques on the ship records. The data that was required to be produced, was to create visualizations on;

- The proportions of individual crew ranks on the ships
- Display the track records of individual sailors
- Histogram of number of crew on each ship
- Number of ship visits to each port, except Aberystwyth

The handbook 'Databases For Historians' by Dr Mark Merry[1], talks about these problems in great detail. Within the 'Designing Databases for Historical Research' chapter of the article he talks about the problems that are faced within historical research and methods that could help to solve them. These problems are;

- Geographical information - This is where the data collected for a certain area can change over time, this could be because the locations were taken over by other countries or destroyed. Meaning data recorded may differ over time, this could be data that represents a location of the town or place. It could also be due to borders changing due to wars over territory or land.
- Chronological/dating information - This is where the different dating systems can alter over time periods and how the person at the time would record the information. This leads to different ways of recording the same date and altered information. An example of this is that you could write 20/05/1998 and in another time period would be Wednesday after x event.

- Orthography/variant forms - This is where you could get data with different spelling or wording but in reality means the same thing. This means that the data cannot be simplified to one meaning easily. The problem even expands to the cases where when recording the data the researcher could not interpret the records or understand what they meant properly. Therefore, this would lead to a misunderstanding within the data.

These problems cause a lot of trouble for historians doing research on the records and causes even greater problems for code to translate the data given into items that could be queried and is still accurate enough to its original source. Dr Mark Merry[1] in his handbook suggests possible solutions to these problems, these are;

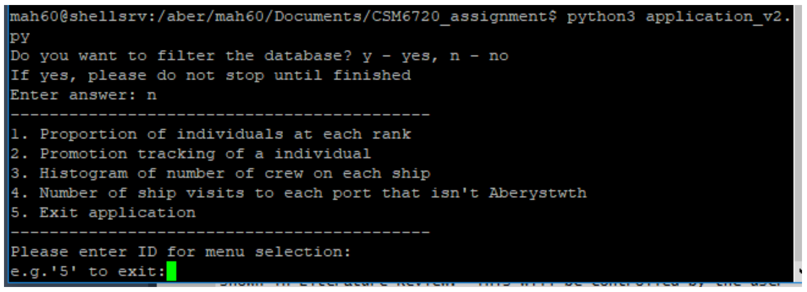
- Standardisation - Method of declaring one form of representation for a set of words that means the same thing. This could be used to filter the data that only looks slightly different from one another and could correct spelling or inaccuracies within the data.
- Classification - This is the process of grouping together a set of data due to some process or theoretical ideology that may indicate relationships between the data. This method is less about capturing the information in your sources and more for serving the researchers needs.
- Coding - The process of substituting one value for another, this could be used to record complex and varied data to a much simpler form and consistent value. This will reduce the processing time for that data.

Overall, applying databases to historical information can be seen as a complex task in that the data can have many missing, complex or mutated elements, this can therefore show an uncertainty it comes to the the processing of the data. The goal of this assignment is to try and overcome these problems and produce results to help visualize the queries that come with the ships records that were asked for.

2 Methodology

The method that was followed was inspired by the research that was discussed by Dr Mark Merry in 'Databases For Historians'[1] and mentioned in the Literature Review chapter. This was to look at the problems that he had discussed and how to reduce the inconsistency of the data within the database. The code that was written to tackle the assignments given to us based on the ship records, were written in python 3.5[3] and talks to the database that is contained in the DBMS of MongoDB[2], which is a document-orientated language.

The code structure that was developed was to first write a script that would filter the database and create a standard within the data. This would then make a duplicate database. This was done to make it easier to process, such as removing unwanted symbols ([,],$,etc) and lowercase string values to make matching data easier. This would also be used to make sure that dates were set to a certain standard and items that were classified to null being set to the value of 'none'. This would therefore reduce the processing that is needed to be done throughout the following stages. You can run this filtering code at the start of application.py by typing 'y' to the first question asked, however, if you do this please let it complete. Otherwise, the code will force the filtering and it takes around two minutes to complete. After a standard has been meet throughout the database for generic data, we can process the data to produce the visualisations asked within the criteria of this assignment, shown in the Literature Review. This will be controlled by the user through input at the menu. These can be seen in figure 1. The other filtering processes that are less generic to all categories within the ship records database are done later on in the program.



```
mah60@shellsrv:/aber/mah60/Documents/CSM6720_assignment$ python3 application_v2.
py
Do you want to filter the database? y - yes, n - no
If yes, please do not stop until finished
Enter answer: n
------------------------------------------
1. Proportion of individuals at each rank
2. Promotion tracking of a individual
3. Histogram of number of crew on each ship
4. Number of ship visits to each port that isn't Aberystwth
5. Exit application
------------------------------------------
Please enter ID for menu selection:
e.g.'5' to exit:
```

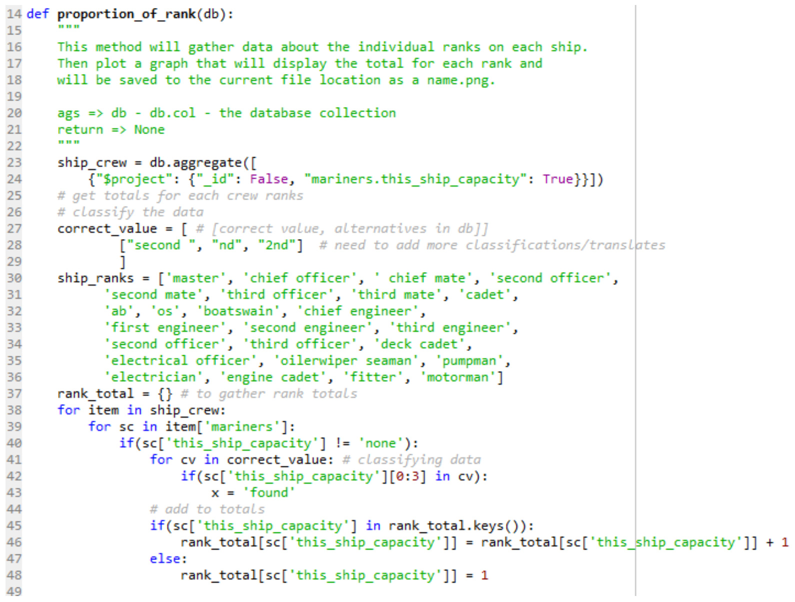
Figure 1: Menu And Filtering Question

The results that were produced from the different parts of the menu can be seen in figure 6. The methods that were used to create these visualisations can be seen in the following subsections of this chapter.

2.1 Proportions Of Ranks

The development of this visualisation was done mostly within the python code. This meant using the aggregate() method on the collections to project only the crew capacities. After this information was gathered, certain filtering techniques were applied against the data to try and translate to one meaning. To classify different wordings of a value to one common wording. This was done by gathering data about different ranks on a ship[4]. Then translating that data to a specific class. This was done to reduce the abnormalities within the ranks, as the different meanings could be '2nd' or 'second' for second mate. Therefore the data needed to be classified to one form.

Then after this was complete the totals for each rank was calculated. To reduce the amount of random ranks that were misspelled or weren't accurate enough to be classified, totals that had a lower value than x were removed. After this the graph was plotted to produce the visualisation seen in figure 6. The code can be seen in figure 2



```python
def proportion_of_rank(db):
    """
    This method will gather data about the individual ranks on each ship.
    Then plot a graph that will display the total for each rank and
    will be saved to the current file location as a name.png.

    ags => db - db.col - the database collection
    return => None
    """
    ship_crew = db.aggregate([
        {"$project": {"_id": False, "mariners.this_ship_capacity": True}}])
    # get totals for each crew ranks
    # classify the data
    correct_value = [ # [correct value, alternatives in db]]
            ["second ", "nd", "2nd"]  # need to add more classifications/translates
            ]
    ship_ranks = ['master', 'chief officer', ' chief mate', 'second officer',
          'second mate', 'third officer', 'third mate', 'cadet',
          'ab', 'os', 'boatswain', 'chief engineer',
          'first engineer', 'second engineer', 'third engineer',
          'second officer', 'third officer', 'deck cadet',
          'electrical officer', 'oilerwiper seaman', 'pumpman',
          'electrician', 'engine cadet', 'fitter', 'motorman']
    rank_total = {} # to gather rank totals
    for item in ship_crew:
        for sc in item['mariners']:
            if(sc['this_ship_capacity'] != 'none'):
                for cv in correct_value: # classifying data
                    if(sc['this_ship_capacity'][0:3] in cv):
                        x = 'found'
                # add to totals
                if(sc['this_ship_capacity'] in rank_total.keys()):
                    rank_total[sc['this_ship_capacity']] = rank_total[sc['this_ship_capacity']] + 1
                else:
                    rank_total[sc['this_ship_capacity']] = 1

```

Figure 2: Proportion of Rank Code

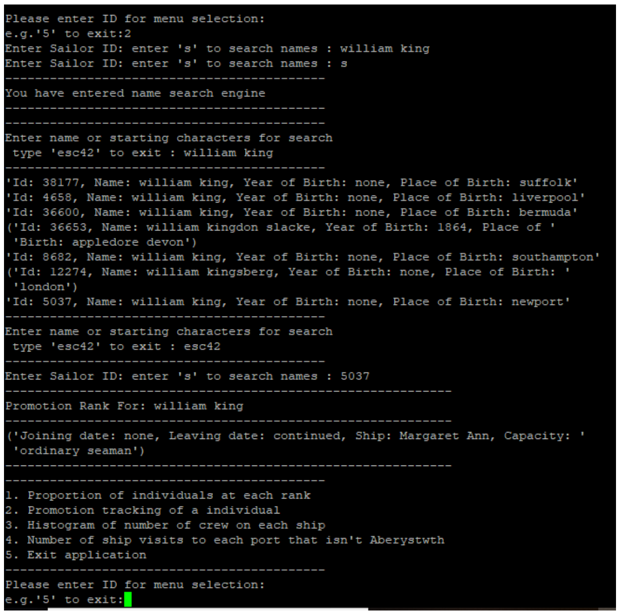
2.2 Promotion Track

The development of this visualisation has multiple stages to it. These are;

- Gather name, place of birth and year of birth. This is because these values together classified the different sailors on the ships. There is no ID to determine who is who, so this was the optimal path to take to get an ID. However, this method is not 100% accurate as there is a chance of two people have the same records still and then would therefore not be unique enough for a primary key value.
- Develop a selection menu, with a search engine. That you can use to type in a name and locate the generated ID for x person.
- Create a promotion timeline/track to view where the x person was on the different stages of his life in x ships.

The first objective was to gather the details about each crew member for the ship records required as an aggregate() method that groups all mariner records that have the same name, place of birth and year of birth was written. This was done to reduce the amount of duplicates within the search menu and allow you to specify a specific person easily.

During the gathering of data for each person a ID was given to help locate that person later and select them within the search menu. To get to the search menu, you must enter 's' on the initial menu and then type the name you want to search. This can be seen in figure 3. When the ID has been entered for the specific person a terminal timeline will be produced and displayed on the terminal.

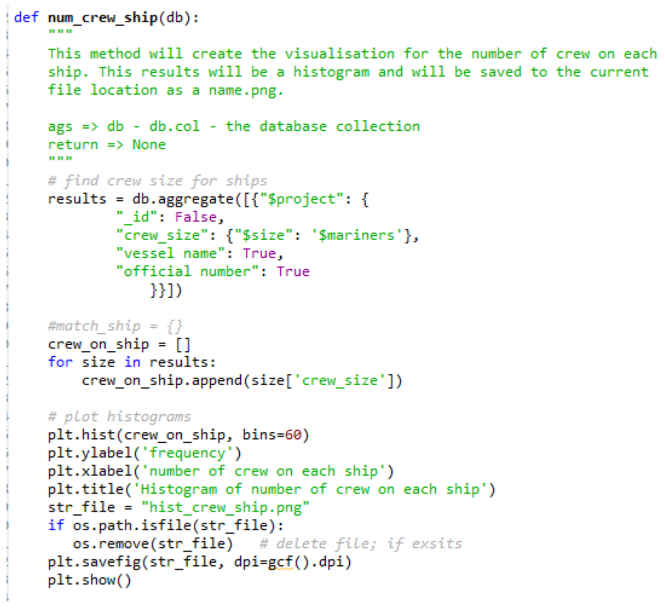


```
Please enter ID for menu selection:
e.g.'5' to exit:2
Enter Sailor ID: enter 's' to search names : william king
Enter Sailor ID: enter 's' to search names : s
-------------------------------------------
You have entered name search engine
-------------------------------------------
-------------------------------------------
Enter name or starting characters for search
 type 'esc42' to exit : william king
-------------------------------------------
'Id: 38177, Name: william king, Year of Birth: none, Place of Birth: suffolk'
'Id: 4658, Name: william king, Year of Birth: none, Place of Birth: liverpool'
'Id: 36600, Name: william king, Year of Birth: none, Place of Birth: bermuda'
('Id: 36653, Name: william kingdon slacke, Year of Birth: 1864, Place of '
 'Birth: appledore devon')
'Id: 8682, Name: william king, Year of Birth: none, Place of Birth: southampton'
('Id: 12274, Name: william kingsberg, Year of Birth: none, Place of Birth: '
 'london')
'Id: 5037, Name: william king, Year of Birth: none, Place of Birth: newport'
-------------------------------------------
Enter name or starting characters for search
 type 'esc42' to exit : esc42
-------------------------------------------
Enter Sailor ID: enter 's' to search names : 5037
------------------------------------------------------------
Promotion Rank For: william king
------------------------------------------------------------
('Joining date: none, Leaving date: continued, Ship: Margaret Ann, Capacity: '
 'ordinary seaman')
------------------------------------------------------------
-------------------------------------------
1. Proportion of individuals at each rank
2. Promotion tracking of a individual
3. Histogram of number of crew on each ship
4. Number of ship visits to each port that isn't Aberystwth
5. Exit application
-------------------------------------------
Please enter ID for menu selection:
e.g.'5' to exit:
```

Figure 3: Search Menu Example & Timeline

2.3 Histogram Of Crew On Each Ship

This visualisation was simple to develop as it required a simple aggregate() method that calculated the size of each mariners array within the ship records and returned that value along with the ship details. This was done as the customer requested specifically for a histogram which would be the frequency of the number of crew on each ship. The aggregate statement can be seen within figure 4, after this the histogram was plotted and saved. This was simple to complete as the data had been filtered earlier on to allow for easy processing.



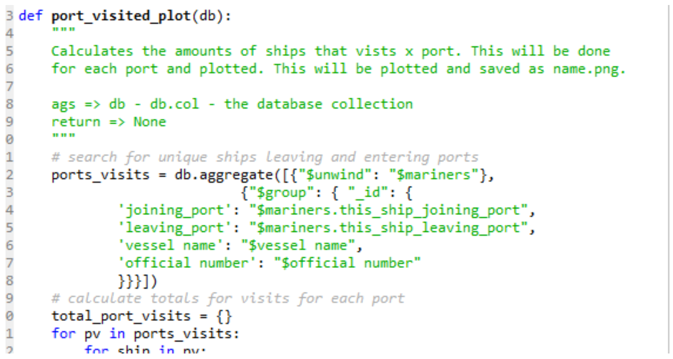
```
def num_crew_ship(db):
    """
    This method will create the visualisation for the number of crew on each
    ship. This results will be a histogram and will be saved to the current
    file location as a name.png.

    ags => db - db.col - the database collection
    return => None
    """
    # find crew size for ships
    results = db.aggregate([{"$project": {
             "_id": False,
             "crew_size": {"$size": '$mariners'},
             "vessel name": True,
             "official number": True
                 }}])

    #match_ship = {}
    crew_on_ship = []
    for size in results:
        crew_on_ship.append(size['crew_size'])

    # plot histograms
    plt.hist(crew_on_ship, bins=60)
    plt.ylabel('frequency')
    plt.xlabel('number of crew on each ship')
    plt.title('Histogram of number of crew on each ship')
    str_file = "hist_crew_ship.png"
    if os.path.isfile(str_file):
       os.remove(str_file)  # delete file; if exsits
    plt.savefig(str_file, dpi=gcf().dpi)
    plt.show()
```

Figure 4: Histogram Code

2.4 Number Of Ship Visits To Each Port

This visualisation took more of a complex aggregate() method to put together, this was done by unwinding the mariners data to view ship joining ports and ship leaving ports of the crew members and grouping them together with the vessel name and the official number. This was done to produce unique data of different visits to individual ports. After this pipeline was created, the totals of visits were added up for each port and sorted to make a top 10 ports that were visited in the time period of the ship records. The top 10 were selected to remove any data that wasn't valid from the final graph. This code can be seen within figure 5.

The code for these visualisation were developed to demonstrate attributes about the original data set and what the ship records can tell us. Methods were developed in each of these visualisation to try and reduce the inaccuracies that can be found within the data. However, these may need to be added upon to further increase accuracy of the results, these are simple to modify and add to.



```python
def port_visited_plot(db):
    """
    Calculates the amounts of ships that vists x port. This will be done
    for each port and plotted. This will be plotted and saved as name.png.

    ags => db - db.col - the database collection
    return => None
    """
    # search for unique ships leaving and entering ports
    ports_visits = db.aggregate([{"$unwind": "$mariners"},
                           {"$group": { "_id": {
            'joining_port': "$mariners.this_ship_joining_port",
            'leaving_port': "$mariners.this_ship_leaving_port",
            'vessel name': "$vessel name",
            'official number': "$official number"
            }}}])
    # calculate totals for visits for each port
    total_port_visits = {}
    for pv in ports_visits:
        for ship in pv:
```

Figure 5: Aggregate Method

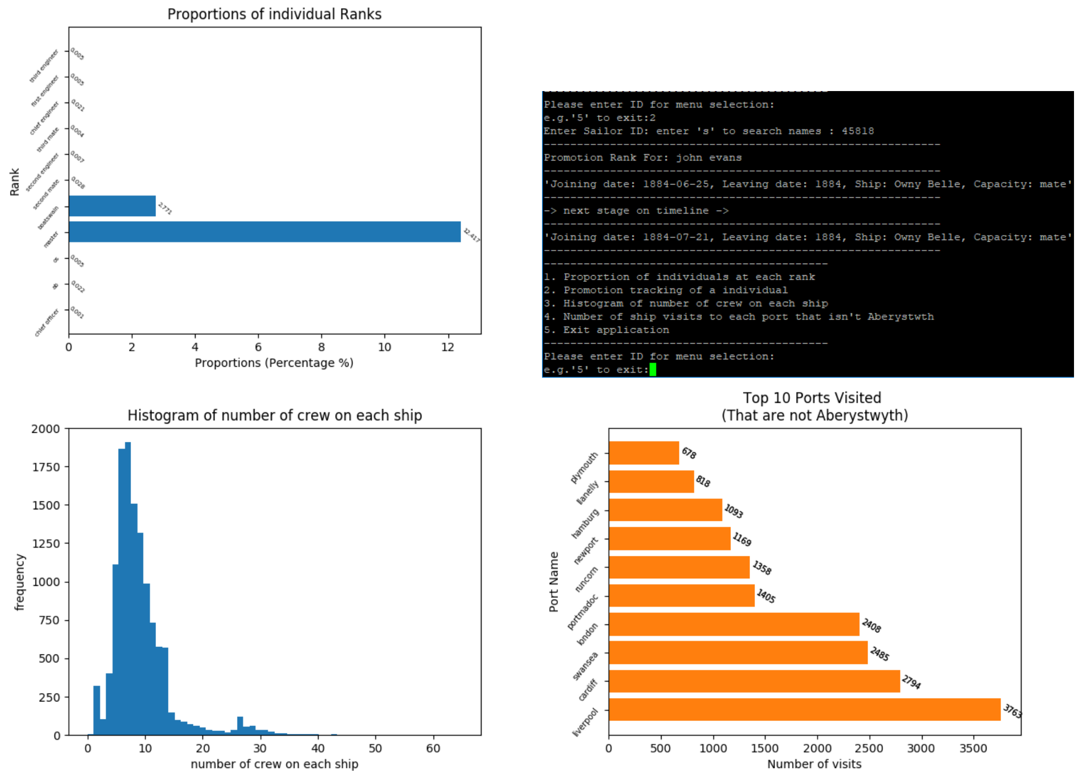
3 Results

The objective of this research was to develop data mining techniques for historical data. The overall results can be seen within 6 and 3, for another timeline demonstration. These visualisations can be produced within milliseconds of entering the value for the menu, however, they have many flaws to them.

The initial process of filtering the data can be seen as a major flaw within the code as it takes around two minutes to complete. Meaning that if you run multiple programs at once this could increase the amount of time to complete the filtering. However, this method has its positives too because doing this overall reduces the run time for the other functions within the program and the filtering system generally only needs to be run once. When the hard code for the filtering method is modified to create a more accurate generic filtering system this will incur extra run turns. The problem with filtering the data at the start of the program is that you could lose the richness and accuracy to the original source. Especially if a mistake is made in the filtering process. This is why the main filtering functions for the individual tasks are in the hard code.

Next to this, the visualisations made have their own perks on the results that were generated. These results were produced at a fast speed, however, there are still errors within some of their methods of filtering to get accurate values. Such as, the proportions of ranks needs to be classified more to produce better results in the different rank categories, because some of the percentages produced are clearly too small for that rank. This can be seen in the 'third mate' rank that has value of 0.004%. Therefore better methods will need to be developed to accurately classify that data. In spite of that a technical success is that the dates for the timeline are sorted correctly. However, if the data value is unknown or does not represent a date, that value is placed at the end of the timeline and is not ordered. On the other hand there has been code implemented to handle multiple different date formats.

The final two graphs for the ships values, are not too complex and only require aggregate() methods to produce. This is due to the development of the duplicate database, which is filtered and means handling the data is easier. However, there are still ships that are misrepresented within the database along with the other variables, these are seen as blank, dates or random characters.



```
-----------------------------------------
Please enter ID for menu selection:
e.g.'5' to exit:2
Enter Sailor ID: enter 's' to search names : 45818
------------------------------------------------------------
Promotion Rank For: john evans
------------------------------------------------------------
'Joining date: 1884-06-25, Leaving date: 1884, Ship: Owny Belle, Capacity: mate'
------------------------------------------------------------
-> next stage on timeline ->
------------------------------------------------------------
'Joining date: 1884-07-21, Leaving date: 1884, Ship: Owny Belle, Capacity: mate'
------------------------------------------------------------
-------------------------------------------
1. Proportion of individuals at each rank
2. Promotion tracking of a individual
3. Histogram of number of crew on each ship
4. Number of ship visits to each port that isn't Aberystwth
5. Exit application
-------------------------------------------
Please enter ID for menu selection:
e.g.'5' to exit:
```

Figure 6: Gathered Results

Furthermore, the code that was developed would struggle to handle large amounts of data. This could be solved using the Mapreduce methods and would reduce the run time of the code[5]. This would be simple to implement into the code as instead of using the aggregate() methods to create a cursor, you could use Mapreduce to create collection with the queried data. The Mapreduce runs in parallel to process and sorts large amounts of data. In spite of that, this was not implemented into the code as the ship records were not large enough for this to make much of a difference in processing time between the two strategies.

Overall, the results demonstrate what was asked for. However, they could be developed on technically to reduce the number of inaccuracies within the data to gain more accurate results but on the other hand the data must also be accurate enough towards the orginal source. The key to this will be to create a balance between the two, rather than rewriting the database for accuracy in the DBMS and not towards the original source. Even though this will increase the processing time, it will be an efficient trade off to the current results, as they will be more accurate.

4 Conclusion

Overall, the assignment was to apply data mining techniques to historical data. This was achieved and visualisations of the different criteria were produced. The attempts show a basic understanding of data mining and how to produce simple results. The overall methodology of the code uses the techniques talked about in 'Databases for Historians' by Dr Mark Merry[1]. The ship records is standardized to a more accurate version of the variables, this was done by removing symbols, letters and numbers when appropriate. There are also classifications within the proportions of ranks function, as the data is being split into the different crew ranks depending on their wording. However, these techniques are at a basic level as there are not enough values implemented to make more accurate data for visualisation. On the other hand, the code within the functions to implement this is simple to add to, for example, entering a value into an array inside the specific function.

The timing was the main limitation of the results for this assignment this was due to time management and learning MongoDB when using python 3.5 took longer than planned to get the structure of the code correct. Another, major problem that took a large amount of time was trying to fix a bug that randomly occurred. This was that when plotting a graph and using the plt.savefig() function, it would bug out and produce what can be seen in figure 7. This bug could be produced from not allowing the matplotlib.plt.save_fig() function enough time to accurately save the file.

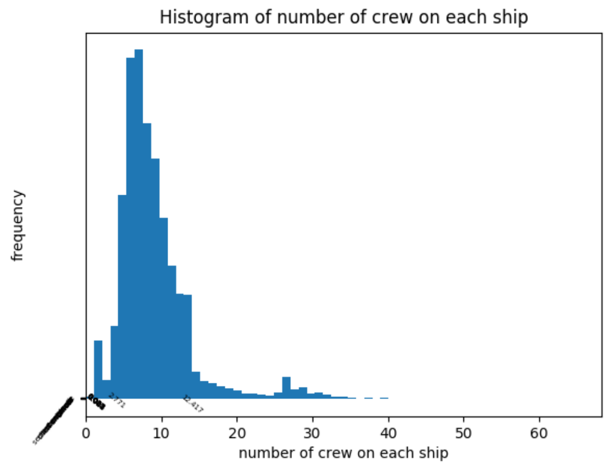


Figure 7: plt.savefig() Bug

As a whole, the developed project and results demonstrate a understanding of how to apply data mining techniques to historical research. Nevertheless the code is robust, to an extent and provides a baseline in which the functions could be developed upon. These methods could be used to make a more accurate set of results and would support future research into data mining techniques for historical data. This is because the current code could be used as a baseline on where to start in future projects.

References

[1] Databases For Historians, Handbook; Dr Mark Merry ;07/05/2019;
https://port.sas.ac.uk/mod/book/view.php?id=75chapterid=148

[2] MongoDB; Database Management System; 07/05/2019;
https://www.mongodb.com/

[3] Python 3.5; Coding language; 07/05/2019;
https://www.python.org/about/

[4] Nedcon Blog; Nedcon Maritime; 01/23/2013;06/05/2019;
http://www.nedcon.ro/crew-structure-on-board-merchant-vessels-deck-department/

[5] Tutorial Point; Map Reduce ; 08/05/2019;
https://www.tutorialspoint.com/mongodb/mongodb_map_reduce.htm