A Toolkit For Reporting On Metagenome Assembly Quality

Final Report for CS39440 Major Project

Author: James Edward Euesden (jee22@aber.ac.uk)
Supervisor: Amanda Clare (afc@aber.ac.uk)

4th March 2016
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name .....

Date .....

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name .....

Date .....

Acknowledgements

I am grateful to...

I’d like to thank...

Abstract

Include an abstract for your project. This should be no more than 300 words.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

1.1 Introduction

The project aim was to create an application that could report on the quality of a metagenome assemblies provided by the user, presenting them with feedback about the contiguous reads contained in their data. The requirements for the project topic were very open, as there are multiple techniques for attempting to report the quality of a single species genome assembly, and while some can be used for metagenome assemblies, it was believed that no one tool covered this area yet with numerous techniques. Likewise, the way in which the results could be presented to the user was not established and open to my own interpretation.

1.2 Background

Before the project began, my knowledge of metagenomics was very limited, close to non. However, I was liked the project title and description and thought it would be an interesting and challenging topic, to learn new domain knowledge, use different technologies and attempt to implement an application where I had to learn from the ground up. On top of this, I find DNA to be an intriguing topic even with my limited knowledge, I was curious to learn more as I worked on this project.

My first step was trying to understand what exactly it was I was expected to produce at the end of the project. As the requirements of the resulting application were so open, it was up to me to research what metagenomics is, what is meant by 'quality' within the subject, how this quality might be found and reported on, what technologies would be appropriate and what quality techniques could be used.

1.2.1 Metagenomics

Metagenomics is the study of environmental samples of genomic data where the contents of the data are potentially unknown and unclear. It can be described as 'Open-ended sequencing of nucleic acids recovered directly from samples without culture or target-specific enrichment or amplification; usually applies to the study of microbial communities.' [4] It can be used in the

findings of what an animal gut may contain, what viruses are within a sample when looking into outbreaks and finding what microbial communities exist in a sample area.

Metagenomics is a hot and interesting topic in the bioinformatics field, and its uses grow as more is learned, but there is the issue of quality, and how a metagenomic sample should be processed. I read articles that attempted to provided ways of analyzing metagenomic data to get the best quality results at the end. [11]

1.2.2 Understanding Quality

Considering the nature of metagenomics and the unknowns, it becomes clear quite quickly that when a sample you may have is run through an assembler in an attempt to create a genome for sequencing, without the proper tools to assess your data you cannot be sure if what you are creating is an actual thing that could exist in nature. The processes of taking a sample through to sequencing with metagenomic data can be very error prone, leading to misassemblies with duplicate or short reads, or combining reads together to make chimeric contiguous reads (contig). [3]

A chimieric contig is an instance where an assembler has put together reads from a sample that it believed were part of the same whole, and yet were in fact of different species/sub-species, and so creates a contig that does not actually exist in nature. It can be understood then that if a user were to sequence this, unless that is the result they wanted, it won't be of any use to them, but without the proper tool how would they know that their assembly data contains chimeric contigs and are not just wasting their time and money?

1.2.3 Existing Software

There are a number of tools I discovered in my background research that attempt to aid in the quality assessment of metagenomic data, in particular MEGAN, a 'next-generation metagenomic data, called MEtaGenomeANalyzer', which attempts to do a taxonomical analysis comparison to known reference data [2] and PRINSEQ, a tool which provides 'summary statistics of FASTA (and QUAL) or FASTQ files in tabular and graphical form' [7]. When considering what my own application should do, I looked at the techniques used for PRINSEQ most, as these seemed to match up with what I thought would be useful to a user for my own application, and from discussing the topic with my supervisor I found techniques such as the GC Content distribution could be a good place to start.

It was not just tools for metagenomic quality assessment I looked into. I also found the NCBI database and their BLAST tool [1], and kept in mind these may be useful as I progressed through my applications development, and through reading an article that discussed the advantages of k-mer frequency analysis for quality assessment, I looked into the Jellyfish [5] and BFCounter [6] tools for just this role where I could potentially consume the output of their processing, although they took a step back in my mind while I considered what it was I actually wanted my application to be and worked out the requirements.

1.3 Analysis

After understanding the project topic and problem a little better, I decided upon a number of requirements of the application to begin with. Some of these were definite goals, some stretch goals and some future development tasks if I were to finish all else or were to continue the application after the project deadline. I broke the problem down into its two core components steps, the analysis and the report.

1.3.1 Quality assessment

The quality assessment had to use a number of methods suitable enough to produce some data or statistics that could indicate to the user a measure of quality of their assembly data. For this end, I decided upon a number of objectives:

1.3.1.1 Contig length

'Give the user control over the minimum length a contig should be to be considered.'

This measure would be good in displaying where an assembler could not find any reads to match with the current read and so didn't do anything more with it than output it. However, this would most likely indicate that it is of no use to a user, as a contig the size of a read length is unlikely to contain any useful genome data. Allowing a user to set the minimum length threshold lets them set their own size, be it the known read length size of their data, or a size they think would be appropriate to start seeing some usable data from contigs with length over a particular amount/number of read lengths.

1.3.1.2 Number of unknown characters

'The application should count the number of unknown N characters within a contig.'

When an assembler cannot understand what to do with a character, or a sequence of characters, it may insert an 'N' character. Indicating to a user how many of these exist in a contig, and what percentage of the whole they make up is a helpful indication of whether their data is of good quality or not, with the less or no unknowns the better the quality.

1.3.1.3 GC Content

'Conduct a GC Content percentage analysis in sliding windows of a size set by the user.'

The GC content is the percentage of 'G' (guanine) and 'C' (cytosine) characters within a sequence [8]. When the application takes a contig, it should break the contig up into windows of a size set by the user. If a contig was of length 30,000, they might break it into 100 windows of 300, for example. These window sizes should then have their GC content percentage calculated, and the mean of the entire contigs GC content worked out. Using these values it then becomes possible to detect potential anomalies in the percentages of individual windows that have a percentage value drastically different than the mean.

1.3.1.4 Open Reading Frame Locations

'To confirm if a GC window is outside of the mean by misalign, Open Reading Frame (ORF) Locations should be found where the GC content would naturally be higher.'

This addition to the requirements came later in the projects development as I came to understand GC content and how a window that looks like an area of a misalign may actually not be. We can look at a contiguous read and find protein coding regions through the use of Open Reading Frames [10], where the GC content percentage is often naturally higher than outside of these regions.

By finding the ORF Locations, we can use these in the quality report to match up to windows that may at first seem like anomalies in the contig where it is instead actually a natural occurrence. There is a preexisting tool for this ORF finding functionality by NCBI [9]. While it might not be the case that an ORF Location matches with an out of threshold GC content window, or that they do match but it is still in fact an anomaly, it was a requirement of the application I wished to include in order to give the user another tool to use for inspecting their data.

1.3.1.5 K-mer frequency analysis

'The application should conduct a k-mer frequency analysis, or use output of k-mer frequency analysis tools.'

Conducting k-mer frequency analysis in window sizes has a similar output as GC content, except we look for the frequency of particular 'k-mer' (where 'k' is a number of how many characters to be considered, e.g. 3mer 'ATG', 4mer 'ATGA'). Through measuring the frequency in windows we could see if there was an even distribution of frequencies across a contig, and if any windows had large changes in the frequencies that could indicate a potentially bad quality contig.

This was set as a stretch requirement, as the process of efficiently conducting k-mer frequency analysis and the research into the possibility of writing my own software to do it or consuming output of another application was expected to take over the time for the project, after my background reading and understanding was completed and the other features previously mentioned implemented.

1.3.1.6 NCBI reference data

'The application should compare the contiguous read data to known reference data in the NCBI database with BLAST.'

Using the known reference data, much like with MEGAN, it would be possible to see if any of the contigs in the user data match to known reference data and so can be considered good quality. This was a stretch requirement though, with the idea more in mind for if the application were to be continued to be developed after the project deadline.

1.3.2 Quality report and data input

1.3.2.1 Read user input in FASTA format

1.3.2.2 Display a list of the contigs

1.3.2.3 User control over GC Content, ORF Location and k-mer frequency analysis paramters

1.3.2.4 Visual report on GC Content

1.3.2.5 Visual report on ORF Locations

1.3.2.6 Visual report of matches between GC Content and ORF Locations

1.3.2.7 Visual report on k-mer frequency analysis

Based on the requirements I had selected, I decided that I wanted to write my own software for each functional component. While solutions for each component exists individually, I felt there would be more benefit to a user if I could produce just my own software to support the functionality over requiring them to user third part software in order to use my application too.

Additionally to this, I wanted the technical challenge of writing my own software, and enabling the application to be maintainable and expanded upon in the future rather than relying on third party support applications in order to process a users input. The process of developing the application from scratch would also give me the opportunity to learn more about the domain and technologies for development than just consuming output of other existing applications, even if achieving the end result would take more time.

While I knew the project would come to a close on development at the end of the semester, I also wanted to envision this as a project that could be taken forward in the future to be further developed and maintained. For this reason it also made sense to develop my own application without consuming output from third party software, excluding the FASTA files from an assembler. If one of these third party applications was no longer available or their output changed, it would render my application unusable or require more future modification to adapt to changes by outside sources.

1.3.3 Objectives

What actual requirements did I come up with for my application, based on my Background reading.

- Produce a visual report on a users input assembly data
- Inspect components of individual contiguous reads in the assembly data
- Include GC Content Percentage in set windows for looking at changes outside of a set threshold that may indicate a chimera
- Include Open Reading Frame Locations within the contiguous reads

- Include comparissons between the GC Content Percentages where there are areas out of a users set threshold and ORF Locations, allowing a user to see areas where there may be not be issues if the GC Content was out of the set threshold but within an ORF Location that may explain the difference.
- Include k-mer frequency analysis in set windows for looking at changes outside of a set threshold that may indicate a chimera
- Allow the user to request contiguous reads under certain lengths to be ignored, and told how many were removed in this way
- Allow the user to select their own sizes of windows for the GC Content Percentages, and allow them to select the minimum length of ORF Locations to ignore any they deem too small for their contiguous read

Include USE CASE DIAGRAM of these things here, and how a user interacts with them.

The list was built up through what would seem possible to be done in the time available, producing an application that had room to grow, for example comparing sections and full contiguous reads with known sequences via BLAST/the NCBI database, while providing useful information to users about their assembly file and the particular contiguous reads within them. The report was decided that while it might not be able to give a definite answer of good or bad quality, it may allow the user to see where there could be issues through highlighting sections of contiguous reads and information about the content of the sequences of them, such as how many characters did the assembly struggle to discern ('n' characters).

After much reading of papers, I decide this list was enough to work on, and additional features would have to be left out. I started working on the list with techniques such as the GC Content percentage in mind as I believed the feature would be a simple technique to include to get me started on the project, and allow me to learn more of the domain as I worked on later features. This included the ORF Location finding later, and I believed would lead to the k-mer frequency analysis feature. I knew that the report was necessary and a core part of the problem. I could see the problem broken down into domain understanding, model data and processing and viewing and reporting, and so I considered these objectives enough to work on for my solution to the topic problem.

1.4 Process

While my project could well follow a plan driven approach, such as waterfall or the spiral model, I believed that an agile approach would be suitable for building it, as I developed the different techniques through breaking them down into tasks. I chose to use Scrum for my framework, and used aspect of Extreme Programming for my daily development cycle adapted to a one person project. I may also have used a feature driven methodology, as I was aware of the features I wished to implement, however, I believed a Scrum approach with weekly sprint iterations would better suit my work flow.

1.4.1 Scrum

I felt I could produce the best work by working iteratively, using Scrum and having weekly Sprints. A Sprint is -what is a Sprint- and it would be useful to the project because -why is a Sprint useful here-. I held my sprint planning and retrospectives every Tuesday after meeting with my supervisor.

The Scrum framework had me taking my initial Objectives mentioned in the previous section and turning them into 'epic' Stories. Stories are -write about Stories and tasks here, including examples of Sprints-

Task break downs helped me focus on what was important, and gauge what needed to be done and how much effort vs time it would take -include image of whiteboard with task here-. The Scrum framework was naturally adapted to a single person project, as there was no external scum master, client or manager, and these roles were all filled by my own motivation and discipline to be fullfilled.

Daily stand-ups with peers, and how this would help. Adapted from normally discussing Yesterday/Today/Blockers with those who could help, instead to discussing in order to improve motivation and sounding boards.

1.4.2 Extreme Programming

Extreme programming -is what? references and explanation here-. Through choosing XP, I found while I couldn't implement every technique, I took the benefits of Test Driven Development and Refactoring

Test Driven Development is ... it would be useful for iteratively building upon my application, and help with refactoring later. TDD helps with the design of the application, in structure and in what needs to be developed next, and helps with time constraints through forcing only what needs to be developed to be developed and no more. - Include examples of some of my tests here, going red, green, red, green-

Refactoring is... With the knowledge that I wanted my application to be something that could be built upon in the future, refactoring was vital for cleaning up the code structure as I built it with TDD, making it maintainable by others in the future, ensuring that the code is readable and cutting out clutter and duplicate code. While it may be seen as a time consuing and unnecessary task by some, when looking at an application life cycle it is vital to a projects success in the long term.

1.4.3 Pomodoro Technique

Additionally, I found it useful to break my development work into small cycles, using the Pomodoro technique (cite needed), in order to increase my focus and productivity while doing work. The aim of the pomodoro technique is to spend 25 minutes with full focus on the work, and then giving yourself 5 minutes to stop and have a break. They are also useful for keeping track of work done and visually seeing how much time has been invested into the project for the day -include image of pomodoro day here, and link to the website- Due to the short length of each pomodoro, it was also seen as an effective method for squeezing in work where there was time. By quantifying time into small slots, it made it possible to consider how much time I might have between other

events in the day and work out where it could be possible to put in a single pomodoros worth of work, instead of perhaps not working because I felt I did not have the time, or working and eating into other daily activities if I didn't block out the time and be aware of when to stop.

1.4.4 Project blog

While developing my application I kept a blog about my project, any milestones I reached or interesting sections. While I did not designate a particular time to blog, I believed it would be a useful tool in reflecting on the work done when writing this report, something for my supervisor to be able to check upon my progress between meetings and give me the time to reflect on the work done and any upcoming tasks to help me mentally process where the project was in its lifecycle against the planned objects.

Chapter 2

Design

Talk about UI, Data importing, The way classes are structured, what languages were used, UML, flow diagrams and how the data flows, why does it flow that way. Use Case Diagram

2.1 Overall Architecture

Evolutionary design. No initial design documents. By using my user stories I knew what tasks needed to be done and considered how would be best to implement them. This lead to the design evolving over time, both in the code structure, classes and user interface. Added functionality as and where needed. Use of prototypes, i.e. ORF Location prototype (show via blog, then show via final result screenshots).

Started by considering what langauge would be appropriate for the processing and display, and at first thought Java would be a good idea. Lead on later to realizing I wanted more than just what Java could do by itself, and chose to make a web serivice, using Java Spring Boot, deployed with Tomcat. I could have instead at this point switched to using Ruby on Rails, but with my prototype code for the GC Content percentage finder done in Java, and without the need for any Active Record, it felt like Java would be the better option. Also portable, as the data processed may be quite large, so Java to allow the application to be run through the JVM on any system (e.g. a Linux HPC) would be appropriate.

OO approach relevant to represent things as Objects. Such as a GC result, ORF Result, etc. Easy to access these items in the browser through thyemleaf as Objects with known names and properties.

Method names follow the convention likeThisSortOfThing and will also try to explain what they do in the method name. Likewise, variables and fields are named such that you can understand what they are without commenting. Attempting to make code that reads for itself what it is. Have JavaDoc for all classes to clarify anything that isn't easily understandable from the code itself.

2.1.1 Choice of technologies

Why Java instead of Ruby, for example? Why use Javascript once I decided to use a web service. Because of Plotly and Canvas? Usability? Accessing data. How about how the data is structured

in Javascript? Why did I select plotly and canvas in particular? Fine level control in canvas, ease of use with Plotly, allowing a user far greater control over viewing their charts than I could provide in the time. Why thymeleaf? very useful for accessing Object data provided by the model. Using fragments for importing sections - Helps remove duplicate code, e.g. for importing the header, and keeping files clean and changing internals of particular charts in their own files, e.g. qualityparameters.html fragment.

2.1.2 MVC Framework

2.1.2.1 Model

Data and structure. ORF sections, GC sections, utility classes and functions, controller access

2.1.2.2 View

Building the view with Javascript and HTML5, using thymeleaf to access data in an OO way. Choice of designing for the view, let it have the data from the Model, passed from the Controller, and how it deals with displaying the data is completely down to the view. Could develop some parts of the view within the code and pass that to the View, but decided it would be better to instead pass the data as is, and let the view decide on the way of display. This allows the front-end and design of the view to be changed in the future without having to go into the internals of the code base and Model itself to change. Keeping the View and Model separate.

2.1.2.3 Controller

Access through GET and POST methods, validation through Java Spring, response to requests and putting objects into the model to be accessed via thymeleaf, through the controller and put into the view.

RESTful interface? GET/POST with session parameters. URL is basic 'list' 'toolkit' 'welcome' represents what you get and addressable when needed. (Remind yourself about the principles of REST and talk about them here!!).

2.1.2.4 User Input

FASTA file format - Standard format for assembly data. Considered File Uploads, and still contains the code for doing this, used for my test files, however, pasting user content was more secure and acceptable for testing. The code for file uploads still exists to be used in revisions to the application. Data is processed in a FASTA reader, and does these things... This way of handling the data, returning it to the controller, and then passing it on to the toolkit area of the model to be 'qualityAssessed', where there are the calls to the techniques used. Again, in a future revision and with more techniques, it may be worth while to allow the user to disable or enable particular techniques they are interested in.

User data is not kept by the application, it is stored in a user session that expires once they leave the page. This is handled by Java Spring, and set as a @sessionvariable (..), show an example

image of these in the code. If the user does not have their session variables, they are presented with the Error page, so they cannot try and access areas of the application/web service where they currently do not have access or the data to do so.

2.1.3 Directory Structure

Java Code stored in three sections. Main is top level, going down into Domain code, Utility code and Web code. All resources, such as stylesheets, html files and javascript stored under resources folder in static directory. HTML stored as template .html files for use by Thymeleaf, Javascript broken up into multiple files based on where they are used (toolkit.js, list.js) and their functions (orf-chart.js, gc-chart.js). Separating out the files this way improves readability and maintainability. Keeping things isolated and simple.

Using MVC - What is MVC - What else did I consider? Did I think about using a command line tool, why no good? Report output. What about using something not a web service? Why did I choose to do a web service instead?

Started working on Java code with the intention of making a GUI that would do the things. Decided

2.1.4 QualitySummary

Results returned in a QualitySummary object. Object contains references to GcResult and OpenReadingFrameResult. Option to have these two implement an interface that might be called QualityResults, and have just an array of QualityResults. This would allow us to add any type of result without needing to know what is in there. However, chose not to do this as considering the way in which we serve the view results and them being so tightly coupled to the view, didn't seem necessary through the evolutionary design. Aware that in the future if we allowed a user to select what type of processing they wish to use, this technique would be very useful to implement, and only serve up the results tabs that they wish to see. However, as stated, evolutionary design through TDD so wasn't yet necessary.

Show how the design is structured. How do we go from the file to the processing? Why did I do it this way? What were my initial designs. Considering ORFS - Result is an ArrayList of the OpenReadingFrameLocations, why this way? First looking at just 'the longest', then prototyping it out to find one in each frame, then breaking it out into finding Start/Stop Codons and 'stitiching' them together where appropriate.

2.2 User Interface

Framework mock ups of the user interface. Logo for kicks.

- Input form, why is it this way? - List - Displaying each contig, why, framework, drop downs - Paramters - Why did I pick these Paramters, what use are these for the user? Why GC content windows and what size is reasonable? Can demonstrate domain knowledge about the choice of parameters. Up to the user, allowing them to put in stupid numbers, but their results are their own. - GC Chart - Why does it look like this - Choice of standard deviation and mean, why? What use

does this actually have in showing the user error areas? What about just visually seeing the data? Show the screenshots of areas out of GC content threshold, show the areas where obvious chimera (50/50 example) - ORF Location - Why did I develop it to look like this? Representing each of the 6 frames, demonstrate the reverse frames are displayed in the correct direction. How did I achieve this and why display it this way and the inner detail? Other approaches? What about the NCBIs version? Ability to click. Also the list? Why have both? They're both useful to have, as a user may want to look at the shortest ORF Location but not be sure where it is even when they read what frame it is. Likewise they may want to see where an ORF Location lies in the list organized into longest to shortest. Why organized into length. Useful to the user to get an idea to further inspect. Why 6 different canvases for the frames? Easier to modify and display on the page. - Superframe - What is it and why did I consider it. Showing overlaps and where the Red areas may be areas of errors and what to look for. The explanation of it, why explain it this way?

2.3 Support Tools

For writing the code, I used JetBrains IntelliJ IDE (screenshot of IDE window with some code). This included writing all the code, Java, HTML, CSS, Javascript, ThymeLeaf and any additional properties for setting up Spring Boot. Useful for text highlighting, debugging the code, and tied into my version control, using Git and hosted on GitHub. (screenshot of commits)

2.3.1 Version Control

Using version control to keep a repository of code in case of lost data in event of hardware crash or software corruption. GitHub decent choice, already had a repository on there and Git supported in IntelliJ IDE. In addition, every time I check in, I used continuous integration with CodeShip, allowing me to receive e-mails any time I made a commit to my repo that didn't pass the tests written. Very helpful in discovering issues with failing builds when checking in a change and forgetting to update tests or breaking previously written code as the design changed (show screenshot of failing/passing builds from CodeShip).

Chapter 3

Implementation

3.0.2 Development Environment

What did I develop it in? My laptop, IntelliJ, Windows 10, PC specs

3.0.3 Features

3.0.3.1 Reading User Input

Implementation What did I do to implement this? Did I do one thing and find it worked for everything? Did I change my implementation. Think about changing from reading in a file to reading user input. Started with a couple of parameters, to big list of parameters, to the UserParameters class (could mention this in Design section too). **Issues** What issues did I encounter, if any? The file names, file sizes being large, formatting of the file and escape characters in user input, symbols in the file? ATGC and default anything else to N. Originally during the reading of the input, would then do the quality assess per contig here, instead moved it to pass back a list of contigs and quality asses on request by controller. Better way of data flow and less class responsibility

3.0.3.2 Counting GC Content & Percentage

3.0.3.3 Displaying GC Content percentage

Prototyped with Plotly. Issues getting data to actually display as I was unfamiliar with plotly and thymeleaf getting data from Objects. Getting the Mean line to show, and setting the colours from within the application - Is this bad? Should probably be done in the View and the model just outputs the data. Decided instead to calculate this in the model itself and output the colours directly. Can still be consumed by parts of the view, and later if a change happens the data is still output

3.0.3.4 Finding Open Reading Frames

Understanding what they were, how they are built. How do I find them in strings? Issues - Finding the longest, how to know when to stop. Originally had the wrong idea and believed all Stop

Codons would be included up until the next Start Codon. Developed against this until feedback from Amanda made me realize the problem was far simpler. Lost developer time, but functionality to fix didn't take too much time. How do we now find ORFs, and store their data? What does it look like, zipping Start and Stop Codons together and getting substrings from within the full contig to create the ORF Location Object.

3.0.3.5 Displaying ORF Locations

Implementation through each ORF Location knowing where it starts and ends, the characters between in a list. Each has a frame indicator, allows them to be kept in a single array. Javascript goes through the list and uses HTML5 canvas to draw them out in 6 different canvases. Working on finding user clicks within the canvas objects, getting data from Thymeleaf to display it from the Objects. Formatting the display and using colour to highlight the start and stop codons. Issues included finding the clicks, highlighting both the menu and the item on canvas. Took some time to get the formatting in the table correct

3.0.3.6 Superframe Comparisson

Making the view of superframe to help the user compare the ORF and GC content. Able to be built upon with k-mer frequency analysis and other techniques as added. The overall 'compare all' chart. Borrowed code from the ORF canvases. Breaking things up into windows from the GC content.

Not really any issues, just trying to find the best way to display it to the user and convey any areas where things don't line up properly. Would like to work on this more and develop a system to detect where the issues really are. Need more techniques.

3.0.3.7 User Interface Organisation

User input section originally having all parameters was messy, only dealing with one contig, then multiple contigs, having parameters appear underneath any contig you wish to inspect, having all charts on the toolbox page displayed in one long list looked ugly, implemented tabs from (credit to the guy here).

3.0.4 Implementation Review

Found that GC Content percentage and ORF Location finding took a lot longer than expected, especially considering that I implemented an additional part of ORF Finding that wasn't needed, and got refactored 3 times until I reached that point. Delayed me to the point I felt I wouldn't have time to implement k-mer frequency analysis. User interface design and programming also took a lot longer. Wanted to present the information in a clean and useful way, without cluttering the page. Getting used to Thymeleaf and how it took the data from the Objects took a while to get used to. Felt that my implementation process was acceptable but issues slowed me down significantly. Everything else I felt I would implement got implemented, but the Superframe needs additional consideration. A lot of the quality assessment is still on the user to look at the produced reports and

their content to consider whether they themselves feel their assembly file is good or not through factors highlighted by my application.

Chapter 4

Testing

4.1 Overall Approach to Testing

Using TDD and refactoring, only did as much development as was required to make tests pass based upon my requriements for each bit of functionality. Also developed test files, both completely artifical to see expected output results, and using real assembly data that I then combined and modified together to result in artifical chimeras to see if I could detect them by looking at the output of my charts.

4.2 Automated Testing

4.2.1 Unit Tests

Used a number of unit tests, writing tests before writing any functional code. Did not write tests for setters and getters, but wrote tests for expected functionality. Made test cases for things such as the ORF Finding (show screenshot of test and tests passing).

4.2.2 Integration Testing

Testing of reading in files. Just asserting the data is as expected, and what happens with blank files, or erronous data files. Does the system handle it gracefully? (Show examples of those test files and what happens with bad files)

4.2.3 User Interface Testing

Tested to see that it worked using Chrome's Developer Tools utility, testing in Chrome, Firefox and Microsoft Edge. (screenshots of how each page looks for each, including results and charts) Testing via a test table (Provided test table) - Tested files with known content and expected results to see response. Tested by viewing what I expected to happen, backing this up with the test table e.g. Click to view information, expected graph results based on artifical test files

4.2.4 Stress Test

Set goal of dealing with large files. Tested with large files to see if the system could process them on my laptop. Could be expanded to deal with much larger files on a system that could handle larger data files with more memory. For my laptop though, dealt with the files fine, even if some took some time to process due to my laptops processing speed. Pasting large amounts of data in a single contig file - Result and time to process file and then inspection time Pasting large amounts of data spread across multiple contigs - Result and time to process file and then inspection time

Chapter 5

Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

Appendices

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

Apache POI library The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the clients existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [**?**]. The library is released using the Apache License [**?**]. This library was used without modification.

Appendix B

Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

Appendix C

Code Examples

3.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [**?**].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*-------------------------------------------------*/
  /* Minimum Standard Random Number Generator        */
  /* Taken from Numerical recipies in C              */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity   */
  /* Always call with negative number to initialise  */
  /*-------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
```

```
static long iy=0;
static long iv[NTAB];
double temp;

if (*idum <=0)
{
  if (-(*idum) < 1)
  {
    *idum = 1;
  }else
  {
    *idum = -(*idum);
  }
  idum2=(*idum);
  for (j=NTAB+7;j>=0;j--)
  {
    k = (*idum)/IQ1;
    *idum = IA1 *(*idum-k*IQ1) - IR1*k;
    if (*idum < 0)
    {
      *idum += IM1;
    }
    if (j < NTAB)
    {
      iv[j] = *idum;
    }
  }
  iy = iv[0];
}
k = (*idum)/IQ1;
*idum = IA1*(*idum-k*IQ1) - IR1*k;
if (*idum < 0)
{
  *idum += IM1;
}
k = (idum2)/IQ2;
idum2 = IA2*(idum2-k*IQ2) - IR2*k;
if (idum2 < 0)
{
  idum2 += IM2;
}
j = iy/NDIV;
iy=iv[j] - idum2;
iv[j] = *idum;
if (iy < 1)
{
  iy += IMM1;
}
```

```
   if ((temp=AM*iy) > RNMX)
   {
     return RNMX;
   }else
   {
     return temp;
   }
 }
```

Annotated Bibliography

[1] N. C. B. I. R. Coordinators, "Database resources of the National Center for Biotechnology Information," *Nucleic Acids Research*, vol. 41, no. D1, pp. D8–D20, Jan. 2013.

The NCBI database and resources was extremely useful to me while developing my application. It gave me access to papers and research articles, data sets for downloading sequences to play with and explore and tools for techniques I wanted to implement in my own application. It's safe to say that without the existence of the NCBI resources, I would have understood a lot less about my project while beginning my development.

[2] D. Huson and S. Mitra, "Introduction to the Analysis of Environmental Sequences: Metagenomics with MEGAN," in *Evolutionary Genomics*, ser. Methods in Molecular Biology, M. Anisimova, Ed. Humana Press, 2012, vol. 856, pp. 415–429.

A useful chapter in reading about a tool that has its own attempt at analysing metagenomic data. While the tool does not look at the quality, it looks at what the user provides and indicates to them if anything they have is already existing in reference data, and how their data compares to similar sequences already known. This could be considered as a quality measure, as if a sequence can be found to match a reference sequence, we know it is most likely not chimeric.

[3] V. Kunin, A. Copeland, A. Lapidus, K. Mavromatis, and P. Hugenholtz, "A Bioinformatician's Guide to Metagenomics," *Microbiology and Molecular Biology Reviews*, vol. 72, no. 4, pp. 557–578, Dec. 2008.

This paper was invaluable to me at the beginning of the project, understanding what metagenomics is, and how the quality of an assembly can be full of errors, and why. While much of the paper was not necessarily relevant to my project as it deals with the full process of working with a metagenomic sample, for my background reading it was very useful.

[4] N. J. Loman, C. Constantinidou, M. Christner, H. Rohde, J. Z. M. Chan, J. Quick, J. C. Weir, C. Quince, G. P. Smith, J. R. Betley, M. Aepfelbacher, and M. J. Pallen, "A Culture-Independent Sequence-Based Metagenomics Approach to the Investigation of an Outbreak of Shiga-Toxigenic Escherichia coli O104:H4," *JAMA*, vol. 309, no. 14, pp. 1502+, Apr. 2013.

This paper was useful in understanding what the study of metagenomics can be used for, and so helped me better understand what metagenomics is and how it could be useful to have a tool for quality control.

[5] G. Marçais and C. Kingsford, "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers," *Bioinformatics*, vol. 27, no. 6, pp. 764–770, Mar. 2011.

Jellyfish, a tool for k-mer frequency anaylsis. A tool I looked into briefly at the beginning of my project when considering what techniques to use for my application.

[6] P. Melsted and J. K. Pritchard, "Efficient counting of k-mers in DNA sequences using a bloom filter," *BMC Bioinformatics*, vol. 12, no. 1, pp. 333+, 2011.

BFCounter, for k-mer frequency analysis. A tool I considered when thinking about the requirements and techniques involved in the development of my application.

[7] R. Schmieder and R. Edwards, "Quality control and preprocessing of metagenomic datasets," *Bioinformatics*, vol. 27, no. 6, pp. btr026–864, Jan. 2011.

A tool that does what my project topic describes, with more understanding than I had at the start of my project. An interesting look at what other software is out there to do what my application also wanted to do and was a baseline for me analysising what sort of techniques I may wish to use to give the user of my own application a report on the quality of their assembly.

[8] J. C. Segen, *The dictionary of modern medicine*. Parthenon Pub. Group, 1992, 1850703213.

For the reference of what GC Content is. The most referenced link seen else where online (e.g. from Wikipedia) is to a web link that no longer exists. This reference backs up the description of GC Content, and is how I understand what GC Content is outside of Wikipedia's reference to the missing web link.

[9] Tatiana Tatusov and Roman Tatusov, "ORF Finder," [Online] Available: http://www.ncbi.nlm.nih.gov/projects/gorf/, 2016, [Accessed on: 25 April 2016].

A tool from NCBI for finding Open Reading Frames within a pasted sequences. I liked using this to compare my own ORF results against and get an idea of how I might want to display my own results in the quality report too. It was also nice to use for helping me understand what an Open Reading Frame actually is.

[10] The Board of Regents of the University of Wisconsin System, "Translation and Open Reading Frames," [Online] Available: http://bioweb.uwlax.edu/GenWeb/Molecular/Seq_Anal/Translation/translation.html, 2008, [Accessed on: 25 April 2016].

I started to understand what an Open Reading Frame was from reading this web page. It helped me get a grasp of what ORFs were, how they were constructed and what I should do to write code for my own application in how to find ORF Locations.

[11] T. Thomas, J. Gilbert, and F. Meyer, "Metagenomics - a guide from sampling to data analysis." *Microbial informatics and experimentation*, vol. 2, no. 1, pp. 3+, 2012.

A useful article in understanding what metagenomic study was. It was helpful to read as I tried to understand metagenomics at the beginning of the project.