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

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

1 Background & Objectives **1**
1.1 Background 1
1.1.1 Metagenomics 1
1.2 Analysis 1
1.2.1 Objectives 2
1.3 Process 3
1.3.1 Scrum 3
1.3.2 Extreme Programming 3
1.3.3 Pomodoro Technique 4
1.3.4 Project blog 4

2 Design **5**
2.1 Overall Architecture 5
2.1.1 Choice of technologies 5
2.1.2 MVC Framework 6
2.1.3 Directory Structure 7
2.1.4 QualitySummary 7
2.2 User Interface 7
2.3 Support Tools 8
2.3.1 Version Control 8

3 Implementation **9**

4 Testing **10**
4.1 Overall Approach to Testing 10
4.2 Automated Testing 10
4.2.1 Unit Tests 10
4.2.2 User Interface Testing 10
4.2.3 Stress Testing 10
4.2.4 Other types of testing 10
4.3 Integration Testing 10
4.4 User Testing 10

5 Evaluation **11**

Appendices **12**

A Third-Party Code and Libraries **13**

B Ethics Submission **14**

C Code Examples **15**
3.1 Random Number Generator 15

Annotated Bibliography **18**

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

1.1 Background

Talk about background stuff here, what did I know about the topic before hand, what didn't I know? How did I prepare for the project - Talking to Sam, talking to Amanda, reading papers on metagenomics, reading papers on quality in genonmics. Show the links to these papers. What was my motivation? Did I find string manipulation and processing interesting, bringing together tools and processes, is genomics interesting?

1.1.1 Metagenomics

What is metagenomics? Why is it interesting. How do we get the assemblies and why are they potentially bad? What makes good and bad quality? Chimeras? Talk about this stuff and show you understand the issues and why this project is something useful to have.

Are there existing system for metagenomics? What about genomics in general? NCBI database and GC/ORF finding tools. k-mer counting as an effective solution and extended task for my project, Jellyfish and BFC with bloom filter. What did I want to achieve? A visual tool for reporting on quality using techniques like comparing GC Content percentage changes and using ORF location finding to see if these areas can be explained or not. Bringing techniques together. Ground work for a tool that can be expanded upon for future ideas.

1.2 Analysis

Analysis of the problem - What is it that needed to be done? What tasks were suitable? Making a user interface with some kind of report using charts and texts. Would it be helpful to do a confidence factor, or leave the user to decide? What sort of interface and data would the user be inputting? Is there a solution to the problem that gives a direct answer? Is it worth writing my own software or consuming results from others software?

My analysis helped to decide that I wanted to write my own software. While solutions exist, their APIs were either depreciated or too much to use, and the output I wished to display would be better when I wrote my own software. 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 alternative approach is the already mentioned method of just consuming output from other applications. While potentially faster to build an application, the risk of the application becoming defunct if the other applications changed the way they output, removed their API or I needed a different form of data meant that it would potentially be a bad idea in the long run, as I wanted to envision this application as something that could have work carried forward and be built upon with additional techniques. It is for this reason I chose to self-implement the techniques, and through that also build on my domain understanding and develop a full application.

1.2.1 Objectives

- 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.

1.3 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.

1.3.1 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.

1.3.2 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.

1.3.3 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.

1.3.4 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.

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 'stiticing' 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. - 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

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 4

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Have you tested your system on real users? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Unit Tests

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

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 [2]. The library is released using the Apache License [1]. 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 [6].

```
#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] Apache Software Foundation, "Apache License, Version 2.0," http://www.apache.org/licenses/LICENSE-2.0, 2004.

This is my annotation. I should add in a description here.

[2] ——, "Apache POI - the Java API for Microsoft Documents," http://poi.apache.org, 2014.

This is my annotation. I should add in a description here.

[3] H. M. Dee and D. C. Hogg, "Navigational strategies in behaviour modelling," *Artificial Intelligence*, vol. 173(2), pp. 329–342, 2009.

This is my annotation. I should add in a description here.

[4] S. Duckworth, "A picture of a kitten at Hellifield Peel," http://www.geograph.org.uk/photo/640959, 2007, copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

This is my annotation. I should add in a description here.

[5] M. Neal, J. Feyereisl, R. Rascunà, and X. Wang, "Don't touch me, I'm fine: Robot autonomy using an artificial innate immune system," in *Proceedings of the 5th International Conference on Artificial Immune Systems*. Springer, 2006, pp. 349–361.

This paper...

[6] W. Press *et al.*, *Numerical recipes in C*. Cambridge University Press Cambridge, 1992, pp. 349–361.

This is my annotation. I can add in comments that are in **bold** and *italics and then other content.*

[7] Various, "Fail blog," http://www.failblog.org/, Aug. 2011, accessed August 2011.

This is my annotation. I should add in a description here.