ELEN4012 - PRELIMINARY REPORT:GENOME ADMIXTURE VISUALISATION

Gary Bezuidenhout 483201
Partners: Isheunesu Jairosi 386233 and Dylan Jacobsen 605262
Supervisor: Prof Scott Hazelhurst

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

Abstract: The aim of this document is to provide insight into the design and proposed implementation of a software tool used to visualise genetic data. The system will be required to generate Principal Component Analysis, Admixture and Manhattan plots. The tool is designed off the basis of the Genesis tool and is required to run as a cross platform desktop application. The implementation of the revised Genesis tool will be written using the Electron framework, visualisation will be developed with the D3.js JavaScript visualisation library. An in-depth analysis of the current Genesis uncovered a number of bugs and inefficiencies. As a result it was concluded that the new tool would be reengineered as opposed to refactored. Having a group of three individuals, rather than two meant that the division of would be critically important, to ensure each member is able to produce sufficient amount of work. A 6 week period has been allocated to complete the project and to ensure this time period is adhered to, a schedule outlining all tasks and estimates has been produced.

Key words: Admixture, Genesis, Manhattan Plot, Principal Component Analysis, Reengineering

1. INTRODUCTION

In the field of population genetics, software tools tend to be over complicated in their operation and in some cases lack certain features to carry out required tasks [1]. Common tools such as eigenstrat, SNPRelate and CLUMPP all produce data output relating to PCA's and Admixture. These tools only produce the mathematical output with regards to the above mentioned PCA and Admixture, however one still requires a tool to plot the data points produced by these systems.

Visualisations are ideal in this case as they produce a visual representation of the data. Data visualisation is beneficial as it generally allows users to gain more insight from data than if they were to look at numbers on a spreadsheet for example. Three benefits are apparent with regards to data visualisation [2] firstly patterns tend to emerge quickly when using visualisations. Outliers are much easier to locate and analysis. Lastly data analysed over time is a much faster process.

The Genesis tool was developed to bridge this gap. It allows users to generate admixture and PCA plots using output from the above mentioned technologies. The tool was developed with user friendliness in mind [1], it allows users to interact with generated visualisations and edit them as required. With the initial Genesis tool achieving its purpose, an updated version is required to remedy the drawbacks of the system. The development of the new application is proposed to correct the drawbacks of the current system while still maintaining the core functionality.

This paper will present the proposed solution of the new Genesis application. Initially project specifications which include constraints, success criteria and assumptions are highlighted. Section 2 presents a background to the problem, this is followed by an evaluation of the current Genesis tool in Section 4. Section 5 highlights the proposed solution for the new application. Sections 6 and 7 contain the Cost Analysis and Project Management approach of the project, these sections are followed by the conclusion.

2. PROJECT REQUIREMENTS AND ANALYSIS

The aim of the project is too evaluate the current Genesis software tool, based on this analysis a new Genesis tool must be developed. The new tool must have the ability to generate three visualisations, namely; PCA, Admixture and Manhattan plots. The project will be considered a success if the application meets the criteria listed below:

- A desktop Application for visualisation of genetic data has been developed.
- The system has the ability to produce three different visualisations, namely the PCA, Admixture and Manhattan plots.
- Generated visualisations must be of high quality and should be interactive, to allow users to modify elements as required.
- Users should have the ability to load genetic data as well as load and save created visualisations.
- Functionality allowing for the download of high quality PDF documents of the generated visualisation(s) should be available.
- Functionality allowing users to label plots.
- Implementation of the application should follow design principles which allow for modularity and extendability.

2.1 Assumptions

- Users of the tool will be limited to academics and students in the field of genetics.
- Despite not having any programming experience users will contain enough knowledge to operate

the application.
- Visualisations must be visually appealing.
- No internet connection is required to use application.
- Users are familiar with PCA analysis of genotypes and have used related software tools. [1]
- All data provided as input for generation of one of three visualisations will be in one of the supported formats.
- No installation required to run the application.

2.2 Constraints

- All implementation, testing and analysis is to be completed in the allotted 6 week time period.
- All code must be well documented.
- Application implementation must be easy to understand.
- The developed tool must be a cross platform desktop application.
- All tools used for the development of the application, should be open source.

3. BACKGROUND

3.1 Population Genetics

Population genetics is a field of biological study that focuses on genetic composition of biological populations and their changes due to varying factors [3]. The field is comprised of two theories present in the genetics, namely Mendelian genetics and Darwin's theory of natural selection [3]. Mendelian genetics is a study that focuses on the process of inheritance, the genetic make up of offspring as a result of the genes passed on by parents [4]. This theory can essentially explain the possibility of a human child possessing blue eyes as a opposed to brown eyes, based on the eye colour and related genes of its parents. Mendel's views on inheritance can be summarized in Mendel's Laws of Inheritance.

Natural Selection is a process by which species within a biological population, adapt to a certain environment. Species that posses certain characteristics which aid in survival and reproduction rate, pass these characteristics onto their offspring [5]. Over time these characteristic will become more apparent across multiple generations. This is how natural selection leads to evolutionary change. It can be seen from the brief explanation of the two theories how their combination, results in the over arcing field of population genetics.

Individuals within the field use mathematical models to determine patterns of genetic variation in actual populations, it involves the analysis of predictable and unpredictable factors. Predictable or deterministic factors include processes such as natural selection, mutation, gene flow and meiotic drive. The primary unpredictable or stochastic driver is genetic drift [6]. By taking into account the external and internal or the deterministic and stochastic drivers, geneticists are able to create models which allow for the analysis of data, which result in a number of useful applications.

There are a wide range of applications brought about from these analyses, within the field of population genetics. In agriculture population genetics is used to understand the evolution of pest which aids in the improvement of crop pest control and in health care population genetics is used to identify how different genes contribute to complex diseases, which aids in human disease control. [7].

There are many tools and analysis techniques used when working with genetic data related to populations. As mentioned above the applications of population genetics are quite vast and tools are required to understand the data.

3.1.1 Principal Component Analysis When working with data sets that include a large number of measurables, analysis can be challenging. In this case some sort of method is required to reduce the meausurables to a more manageable size, while at the same time all important information must be retained. This is where the technique of principal component analysis (PCA) is effective.

PCA is a method for exploring data sets with a large number of measurables by reducing the measures to a smaller number of principal components that highlight the main information [8]. The information extracted when using PCA is represented as a set of new orthogonal variables known as principal components. There are four main requirements associated with PCA[9], these are listed below;

- PCA is required to extract the most important information from the data set.
- The data set is to be compressed by only keeping this important information.
- The description data set must be simplified.
- The structure of the observations and variables must be analysed.

The principal components are ordered according to variance with the highest variance being equivalent to the highest order of importance. Linear algebra is used to determine these orders of importance which in turn provide the principal components related to the accompanying data set. The first principal component or the component of highest importance accounts for the largest amount of variability in the data [8], all subsequent principal components relate to the next highest value of variability.

In the field of population genetics, PCA is used as an analysis tool in studies of human migration. Principal

components generated from PCA carried out on human genetic variation are superimposed on the geography of sampled populations and this generates synthetic maps. These synthetic maps are used to illustrate historic human migrations[8]. This is only one of the many uses of PCA in general as well as in the field of population genetics.

3.1.2 Admixture Generally closely related populations that were isolated from each other are brought into contact as a result of a disturbance in the habitat.This disturbance effectively removes the barrier that caused the isolation between these two populations. When these populations are brought into contact interbreeding occurs and the offspring contains genes passed from both parents, this is known as admixture [10]. Generally in the field of population and statistical genetics one is interested in finding admixture proportion, this information allows individuals to understand the contribution of each population to their genetic make up.

One of the applications of determining admixture proportion is that off admixture mapping. This is a powerful gene mapping technique to determine genes or traits that could be of potential risk passed down from ancestry [11]. Admixture mapping has mostly been applied to the African American community whose ancestry can sometimes be traced to Europe and West Africa. In most cases admixture mapping is used to identify susceptibility to certain genetic diseases based on ancestry. Darvasi et al. [12] presents how the African American population were susceptible to hypertension using admixture mapping.

3.1.3 Manhattan Plot A Manhattan plot is a type of scatter plot that is generally used to plot data that contains a large number of points. In the field of genetics, Manhattan plots are generally used in genome-wide association studies (GWAS).

GWAS is a technique that requires the scanning of complete sets of DNA of many individuals to find genetic variations in particular diseases [13]. Once similarities and differences are identified, individuals can be treated according to their specific case based on the genetic research. This method is laying the foundation for an era of personalized medical treatment.

3.2 Design Architecture Patterns

3.2.1 Model-View-Controller This design pattern splits a software application up into one of three roles, namely; Model, View or Controller (MVC) [14]. Each one of these three components is separated by abstract boundaries and communication between them occurs across those boundaries. Figure 1 illustrates the relationships between the three components that make up the design pattern. The responsibilities of the components that make up MVC are defined below:

Model: the models represent knowledge [15]. They can represent a single object or a structure of different objects. When looking at a model there should be a one to one correspondence between the model and its properties withing the system on one hand and the object in the real world on the other.

Views: the views act as a visual representation of the models [15]. It can be considered as the presentation layer of the data. The views attain the necessary data required to display the knowledge possessed by the model by sending messages to the model.

Controller: the controller is the means for the user to communicate with the system [15]. It allows the user to provide some sort of input and interact in some form with the views created by the models. The controller interacts with the views which in turn communicate with the models to the provide information to the user.

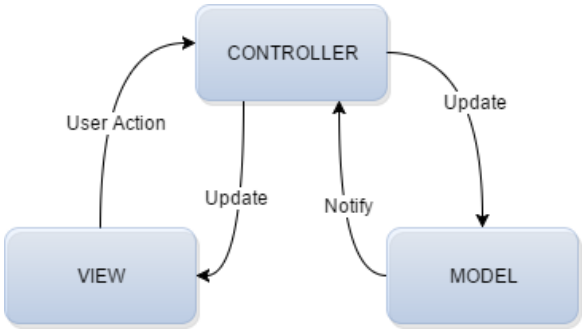


Figure 1 : Model View Controller Design Pattern

The major benefits of the MVC design pattern include the fact that it allows for modular system design as well as easy extendibility due to the distinct separation between components. Reusable code is another benefit of the MVC pattern.

3.2.2 MicroKernel There are many applications that possess a core set of tasks, these tasks are used continuously depending on the requirement of the system. When using a pattern such as this one, a core set of tasks must be predefined in order to ensure that these tasks are prioritised. Plug-ins can be included on top of these core tasks which would allow some kind flexibility in the execution of the tasks. An example of this could be insurance claims, the core process of processing a claim is always the same, however there are slight differences based on the situation. To apply this pattern, the core set of tasks must be pushed into the microkernel and plug-ins would be added on top this based on specific requirements [16].

One difficulty in using this pattern is determining the basic set of tasks that will be used to make up the core

set. Another drawback would be the large number of conditions required to be implemented to ensure that the right plug-in is used in the right situation. The pattern works well in systems that have a well-defined set of core tasks that can be pushed to the microkernel. Systems with a clear distinction between core functionality and features could also work well.

3.2.3 MicroService This pattern works well for small applications but the larger the application the more inefficient the pattern becomes. The goal of micro services is to build an application made up of smaller self-sustaining applications rather than one large application[16]. Every time a new feature is to be included a new program is created. Many different task can be included regardless of the processing power required or the use of the task, this is one of the main benefits of using the micro services design pattern. The converse of this however is that when the system is made up too many subsystems performance can be greatly affected.

3.3 Software Design Methodologies

3.3.1 Waterfall Methodology With its origins in the manufacturing and construction industries, the waterfall design methodology is a sequential design process made up of different stages [17]. These stages include requirements gathering, analysis, design, implementation, testing, deployment and maintenance. Due to the fact that this is a sequential design process each step must be completed before moving onto the next step. It is a very rigid process as there is no room for change or error. There are multiple different types of waterfall methodologies implemented, commonly modified by development teams based on their unique requirements. In some cases, modified waterfall methods include a verification and validation step at the end of each stage to ensure that the developed product does not deviate from the needs of the client. If verification and validation at the end of the stage is not satisfied the team is required to backtrack until this test is passed. This feedback mechanism included in the waterfall method combats one of the main downfalls of the original waterfall methodology [18].

Some of the main benefits of the waterfall methodology is that its simple to follow due to its strict rigidity and it allows for simple testing and analysis of the system. The main drawbacks of the system is that it only works for systems that have very precise requirements and does not work well for long/ongoing projects.[19]

3.3.2 Agile Methodology The agile methodology came about as a solution to the drawbacks of the waterfall methodology [17]. Rather than following a strict sequential process, agile focuses on an incremental approach that aims to deliver value to the client as quickly as possible [20]. Each increment of work is completed in a pre-defined period of time known as a sprint, at the end of each one of these sprints the team must produce a fully functional reduced version of the entire product. This ensures that the client can give consistent feedback to ensure that the final product effectively meets the requirements. One of the main advantages of the agile method is that changes can be made to the system after the initial planning stage, this flexibility assists with projects whose requirements are susceptible to constant change, whether due to the client changing their minds or external factors. The disadvantages of this methodology is that if the project is not managed correctly it can easily go over budget or over time and this methodology lacks documentation efficiency [19].

3.3.3 Xtreme Programming Methodology Xtreme Programming (XP) is a pure agile methodology that takes the essence of agile to its extremes [21]. XP teams are generally small and individuals work in pairs with one programming and the other ensuring that the written code meets team coding standards. This ensures that coding and code reviews are done simultaneously [21]. Planning sessions are generally short as they focus only on immediate functionality required, with the assumption that the future is unclear and change is inevitable, grand planning sessions are not considered within this methodology. Xtreme Programming is based on five main principles: Rapid feedback, assumption of simplicity, incremental change, embracing change and quality work [21]. The main advantage of this methodology is a large focus on customer involvement. Disadvantages include the effectiveness of the project is totally dependent on the team and the need for frequent meetings could raise project costs. [19]

3.4 Software Tools

3.4.1 JavaFX A java based software platform for building and delivering desktop applications. JavaFX is a set of graphical and media packages that allows developers to create and deploy rich client side applications with functionality across multiple platforms [22]. JavaFX applications can reference a range of APIs from any Java library. The visual layer of a JavaFX application is created using CSS. One of the key features of JavaFX is known as WebView, this feature allows developers to embed a web browser into the JavaFX desktop application [22].

3.4.2 Electron Developed by GitHub, electron is an open source tool which allows developers to build cross platform desktop applications using HTML, CSS and Javascript. This is done by combining Node.js and Chromium into a single runtime which brings about

the ability to create apps that can be packaged for Mac, Linux and Windows. [23]

3.5 Visualisation Libraries

3.5.1 D3.js An open source JavaScript library for creating and manipulating documents based on data. D3 allows for high level interactivity of visualisations using HTML, SVG and CSS. D3 provides users will the all the capabilities of modern web browsers without any constraint on the use of frameworks. The library combines powerful visualisation components with a data-driven approach. [24]

3.5.2 JFreeChart An open source Java library used to display charts in Java applications. The library has a well-documented API supporting a wide range of chart types and it contains support for server side and client side applications. The application is compatible with Java Swing and JavaFX applications, it also provides chart outputs in the form of PNG, JPEG and PDF files among others. [25]

3.6 Reengineering and Refactoring

3.6.1 Reverse engineering In [26] reverse engineering is defined as the process of analysing a system to;

- identify the systems components and their interrelationships and,
- create representation of the system in another form or at a higher level of abstraction.

Based on the above definition reverse engineering can be considered purely as a means of inspection. The insights achieved from reverse engineering are one of the initial steps when considering both refactoring and reengineering.

3.6.2 Refactoring/Restructuring Restructuring is the transformation from one representation form to another at the same level of abstraction, this is done while still preserving the systems external behaviour [26]. Essentially restructuring can be referred to as the redesign of a system , without changing what the system does. Transformation of messy-code to structured code can be considered restructuring. Refactoring is changing code to improve its design, it is essentially the same as restructuring however it is concerned with improvement rather than just maintenance [27]. It improves the current system without adding functionality. The second law of software evolution says that software becomes more complex over time, as a result one is forced to apply refactoring/restructuring to keep software maintainable.

3.6.3 Re engineering Software reengineering is the reimplementation of a system (legacy) to make it more maintainable [28]. Reengineering can include redocumenting and refactoring the current system [28]. In essence the entire system is reimplemented , however the functionality remains the same. In some cases when a system undergoes reengineering, some additional features are included. The software reengineering process is made up of five steps, these are highlighted and described below.

- Source code translation: conversion from an old programming language to a modern one.
- Reverse engineering: information is extracted from the program once it has been analysed. This assists in highlighting its organisation and functionality.
- Program structure improvement: the control structure is modified to simplify reading and understanding.
- Program modularisation: similiar parts of the program are grouped together and redundancy is removed.
- Data reengineering: data processed by the system must be modified in line with all other modifications made to the program.

There could potentially be costly delays when introducing new software into a business, where as reengineering could either reduce the delay or negate any kind of delays. Thus by undergoing reengineering as opposed to the use of a new system can result in reduction of risk, this is due to the fact that the time delay brought about by the use of a new system could have a negative on the business. With no delay present with reengineering, the risk of these possible negative effects is reduced or negated. Another aspect of introducing a new system will be cost, generally reengineering would be the cheaper option. From the above it can be seen that the main advantages of reengineering are reduced risks and cost [28].

4. EXISTING SOLUTIONS

Applications such as eigenstrat, Admixture, CLLUMP and SNPRelate all produce numerical output, this data is required to be visualised [1]. Due to the lack of simple, easy to use applications, the Genesis application was developed. The aim of the application was to plot admixture and PCA plots, with the added functionality of interactivity to allow users to edit visualisations.

Aside from the core functionality of the system, the system contains a number of additional features, these are highlighted below:

- Appearance Editing: The user has the ability the places labels and lines on the generated visualisation to add extra information. Editing is also available, the user can change details of the graph such as font headings, label positions etc.

- Export Functionality: The user is able to download high quality plots in SVG, PDF and PNG formats
- Manipulation of Samples: The user is able to view selected samples, samples not required can be hidden when viewing the different plots.
- Multiple plots: Multiple admixture plots can be loaded simultaneously for ease of comparison.

Despite the application meeting the basic requirements,there a number of drawbacks in the system. In the Genesis user manual a bug relating to the SWT menu is highlighted, the application has to be opened at least twice for the menu to appear. When running the application the added lines and labels are not dynamic, if some change is made to the plot, the labels or line do not respond to this change. When rotating the 3D PCA plots the grid lines tend to warp. Resizing of the application also tends to be an issue as the plots can sometimes be too large for the screen.

In terms of the software architecture design, there doesn't seem to be a clear design to the structure of the code. When considering the DRY and KISS principles, the current genesis application, has a great deal of repeated code as well as unnecessarily complicated code. One example of this is the discrepancy between the PCA code and the Admixture code, one is much more dense then the other. As previously mentioned, a known bug due to SWT was highlighted in the application documentation [1].

5. PROPOSED METHODOLOGY

The following sections presents the proposed solution for the application. Figure 2 illustrates the overview of the proposed system.

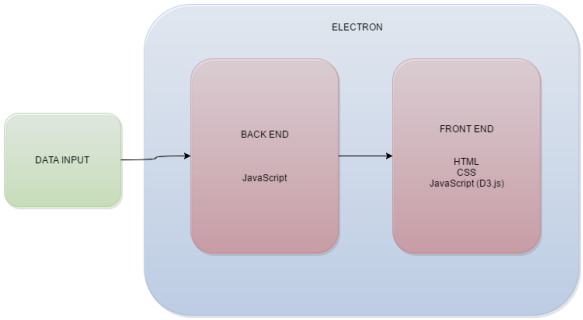


Figure 2 : Proposed Solution System Overview

5.1 Reengineering vs Refactoring

Based on the evaluation of the current Genesis system given in section 4, both refactoring as well as reengineering could be options with regards to the implementation of the new system. However considering the fact that the issues with the system lie in the architecture, implementation and the final product itself, refactoring could become a fairly large task.

Due to the given time constraint of 6 weeks, refactoring may not be a viable option, as a result the group will move forward with the reengineering option. It would make more sense to build the new application from scratch as it would be the option that requires the least amount of time.

This would be a more efficient method in combating all the inefficiencies and issues with the current application.

5.2 Design Methodology

A modified version of the waterfall methodology will be adopted for the development of the application. This modified methodology will include a verification and validation step at the end of each stage of the development process. The team must ensure that all required verification and validation tests are passed before moving onto the next stage of development.

5.3 Design Architecture Pattern

The MVC design pattern will be used for implementation of the application. This design pattern remains consistent with the node.js web application which is based on MVC as well. Models, Views and the controller is easily indetifiable based on the project requirements. The models will posses attributes that map directly to different components of the visualisation. In essence a model would represent each on of the three visualisation. The views will relate to the actual display of the visualisation, while the controller will be the user interaction with the system, to create, edit and download visualisations.

5.4 System Framework

One of the main constraints of the project is the fact that the project must produce a cross platform desktop application, this brought about many limitations in terms of what could be done with project. With this in mind three possible options based on programming language were established. A brief description of these follows:

5.4.1 JavaScript Considering the fact that there is an extensive number of JavaScript visualisation libraries that allow for great deal of interactivity, the use of JavaScript for the front end of the application is preferred. One would generally associate JavaScript with web development, however a software tool known as Electron (discussed in Section 3.4.2) allows for the use of JavaScript to develop cross platform desktop applications. The application code is written in the form of a Node.JS web application and is packaged as a Mac, Linux or Windows desktop application. An added benefit of this method is the fact that the development of a web application would only require the

migration of the Node.JS code, rather than writing an entire web application from scratch. Based on the above information the initial option with regards to the system setup would be the use of Electron to write the desktop application in JavaScript.

5.4.2 Java/JavaScript JavaFX (discussed in section 3.4.1) WebView is a feature that allows developers to include an embedded browser within a java written desktop application. With this in mind the second option with regards to the system setup would be to run a JavaFX application with WebView enabled, the embedded browser would run a page that produces the front end containing the visualisations written in JavaScript, HTML and CSS. A web server would have to be instituted to run the web pages for the front end of the system.

5.4.3 Java The final option remains consistent with the original application with one slight difference. Rather than using SWT, a JavaFX application will be written with both the front end and back end of the system written in Java.

5.4.4 Proposed Solution and Justification The main reason the group has decided to use JavaScript ahead of Java for the development of this tool is for the visualisations. This application is mainly based in the front end and by using JavaScript many of the drawbacks experienced by the initial Genesis application can be remedied with the use of JavaScript. Firstly grid line warping and dynamic resizing are all simple tasks to handle in JavaScript and specifically D3.js. This is not to say that Java is unable to remedy the initial drawbacks with modification to the initial code base, but what is being said is that JavaScript is more effective. The JavaScript visualisation libraries provide a large number of graphing templates and if the required visualisation type cannot be found, D3.js can be used to build the visualisation from scratch, its simple and easily configured to suit the input data. One of the main advantages of using D3.js is it removes constraints with regards to graphing in general, one has freedom to essentially do whatever it is they need to do. Interactivity is one of the most important features of the Genesis tool and will be at the forefront of the new application, D3.js is centered around interactive visualisations.

D3.js is already being used by a number of commercial entities for their visualisation needs. Some of the examples include twitter voting visualisations, visualisation of sports data and visualisations on fashion sites to name a few.

As mentioned above the system will be using the Electron framework, this tool allows users to write their application as if it were a web application in the form of node.js. With the chosen design pattern being MVC, this solution links up effectively with all the design decisions.

One of the requirements of the system is modularity and extendability, by using the proposed JavaScript setup this can be easily achieved.

Finally, an additional feature that can be added alongside the desktop application is a web application. The setup of this web application will be quick and easy as most of the groundwork will be done within the desktop application. This also creates consistency across the desktop and web and also allows users to access the system when they do not have access to their own computers.

Based on the detailed justification given above, the initial option of an Electron framework written in JavaScript will be used for the new Genesis application.

5.5 User Interface

HTML and CSS with the aid of the Bootstrap library, will be used to develop the user interface of the application. There will be 4 main views of the system, descriptions of these views follow:

5.5.1 Start Page The user is able to load existing visualisations or create a new visualisations from this page.

5.5.2 Project Page This page allows users to view and interact with existing visualisations, users can view multiple visualisations simultaneously.

5.5.3 New Visualisation The user is required to set the attributes of the new visualisation.

5.5.4 Data Input Data input selection for generation of a new visualisation.

The views will interact similiar to the interaction of the current system. The user will navigate from the start page and either load an existing project or start a new project. When starting a new project the user will be directed to the new visualisation view which will allow the user to input information pertaining to the new visualisation. There after input data will be selected via the data input view and thereafter the new visualisation will be displayed on the project page. If from the start page the user selects load existing project they will be directed to the project page.

The project page acts as the central hub of the tool, this is where all editing of visualisation will occur, users will have access to all their saved visualisation

from this page and they will also have the ability to view multiple visualisations at a time. An illustration of the interaction between views can be seen in Figure 3.

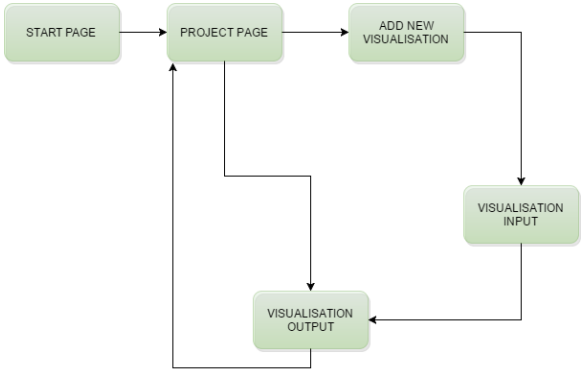


Figure 3 : User Interface Views Interaction

5.6 Visualisations

Based on the decision to use JavaScript as the front end language, D3.js will be used to develop the visualisations. Development is not restricted to D3.js, there are a large number of JavaScript visualisation libraries, however this specific library has an extensive functionality to ensure that all three visualisations can be built to the quality standard required. If required visualisations cannot be found within the D3 archive, D3 will be used to build the visualisations from scratch.

5.7 Testing

For testing, general software practises will be used to test the code written by each member. In terms of the back end, unit testing will be done on the data models and their accompanying functions. Acceptance tests and integration tests will also be undertaken in terms of the back end of the system. The above mentioned testing methods can also be applied to the front end, in addition methods such as visual regression testing and performance and accessibility testing can be carried out. [29]

6. COST ANALYSIS

A comprehensive cost analysis has not been carried out, this is due to the fact that the project is purely software based and all the selected technologies are open sources. Based on the current proposed solution the group requires no funds to develop the application.

7. PROJECT MANAGEMENT AND PLANNING

7.1 Group Management and Work Division

This project will be carried out with three individuals, as such they will have to demonstrate the ability to cooperate and work with one another effectively to ensure that the laboratory project will be a success. This in turn is an invaluable skill which all the individuals would require when undertaking engineering projects in industry. Due to the fact that this project will be done by three individuals rather than two, ensuring that each individual has contributed distinct and meaningful effort to the project is important especially when considering that projects options were designed with a pair in mind.

Considering this the project was extended to three visualisations rather than the initial two stated by the project description. By including a third visualisation it ensures all members can equally add to the project, this also allows the group to work on the system framework together. By splitting the work in this way, each member is exposed to the different parts that make up the project which gives each individual a more complete experience working through this project.

The project has been broken down into 6 subsystems, namely pre planning, system framework, admixture, pca, manhattan and general. All members will contribute to work within each subsystem except those specific to the different visualisations. Table 1 in Appendix A highlights the individuals responsible for each project task. The pre planning stage focuses ensuring all final design decisions and any setups for the project are complete.

The pre planning stage is set to begin a week before the official start date of the project. This will ensure that on the 09 September 2017, the official start date of the project, the group can start actual project work. The Gantt Chart in Appendix A highlights all estimated dates of completion for each task, the group must ensure these estimated dates are adhered to as best as possible to ensure successful completion of the project on the due date.

7.2 Resources

The resources available in this project are mainly human resources which are the three members of the project group. Allocation of these resources, as previously mentioned is illustrated in Table 1, Appendix A. Other than human resources the only other resources required would be a computer with the above mentioned technologies installed for each group member.

8. CONCLUSION

The design of a software tool used for the visualisation of PCA, Admixture and Manhattan plots has been proposed. The design is based off of the current Genesis visualisation tool. The tool must be a cross platform desktop application, to ensure this condition is met the group will implement the new tool using Electron. The new tool will be implemented purely

in JavaScript with the use of D3.js to produce the visualisations. After carrying out an evaluation of the current Genesis tool a large number of inefficiencies were discovered across different aspects of the system. Due to this discovery and time constraints the new design wILLd be reengineered rather than refactored. The proposed design will be implemented in a 6 week period. To ensure this time constraint is met a schedule for implementing the design has been drafted, with the work split equally between the three group members. An Additional features which is building a web application alongside the desktop application is discussed.

REFERENCES

[1] R. Buchmann and S. Hazelhurst. *The Genesis Manual*.

[2] N. Shamas. "Why Data Visualization is Important.", 2015. URL `https://www.techchange.org/2015/05/19/data-visualization-analysis/`.

[3] S. Okasha. "Population Genetics." In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2016 ed., 2016.

[4] G. Generation. "Mendelian Genetics.", 2015. URL `http://knowgenetics.org/mendelian-genetics/`.

[5] D. Futuyma. "Natural Selection: How Evolution Works.", December 2004. URL `http://www.actionbioscience.org/evolution/futuyma.html`.

[6] B. U. Biology and Medicine. "Introduction to Population Genetics." URL `http://biomed.brown.edu/Courses/BIO48/6.PopGen1.HW.drift.HTML`.

[7] J. Pritchard. "Medical Applications of Population Genetics." Department of Statistics, University of Oxford, 2008.

[8] D. Reich, A. L. Price, and N. Patterson. "Principal Component Analysis of Genetic Data." In *Nature Genetics*. Nature Publishing Group, volume 40, issue 5 ed., 2008.

[9] H. Abdi and L. J. Williams. "Principal Component Analysis." In *Interdisciplinary Reviews, Computational Statistics*. Wiley, volume 2, issue 4 ed., 2010.

[10] D. J. Balding, M. Bishop, and C. Cannings. *Handbook of Statistical Genetics*, chap. 30, pp. 1041–1042. UK: Wiley Inc., third ed., 2007.

[11] D. Shriner. "Overview of Admixture Mapping." In J. L. Haines, editor, *Current Protocols in Human Genetics*. John Wiley and Sons, Inc., 2013.

[12] A. Darvasi and S. Shifman. "The beauty of admixture." In *Nature Genetics*. Nature Publishing Group, volume 37, issue 2 ed., 2005.

[13] N. H. G. R. Institute. "Genome-Wide Association Studies." URL `https://www.genome.gov/20019523/`.

[14] A. Developer. "Model-View-Controller." URL `https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html`.

[15] J. Atwood. "Understanding Model-View-Controller." URL `https://blog.codinghorror.com/understanding-model-view-controller/`.

[16] P. Wayner. "The top 5 software architecture patterns: How to make the right choice." URL `https://techbeacon.com/top-5-software-architecture-patterns`.

[17] base36. "Agile and Waterfall Methodoloigies - A Side by side comparison." URL `http://www.base36.com/2012/12/agile-waterfall-methodologies/`.

[18] H. van Vliet. *Software Engineering: Principles and Practice*, chap. 3, pp. 48–50. UK: Wiley Inc., third ed., 2007.

[19] K. Jamsheer. "12 Best Software Development Methodologies with Pros and Cons." URL `http://acodez.in/12-best-software-development-methodologies/`.

[20] H. van Vliet. *Software Engineering: Principles and Practice*, chap. 3, pp. 50–51. UK: Wiley Inc., third ed., 2007.

[21] H. van Vliet. *Software Engineering: Principles and Practice*, chap. 3, pp. 61–64. UK: Wiley Inc., third ed., 2007.

[22] M. Pawlan. "What is JavaFX?" URL `http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm`.

[23] GitHub. "About Electron." URL `https://electron.atom.io/docs/tutorial/about/`.

[24] M. Bostock. "D3 Data Driven Documents." URL `https://d3js.org/`.

[25] D. Gilbert. "JFreeChart." URL `http://www.jfree.org/jfreechart/`.

[26] E. Chikofsky and J. Cross. "Reverse Engineering and Design Recovery: A Taxonomy." In *IEEE Software*. IEEE, volume 7, issue 1 ed., 1990.

[27] H. van Vliet. *Software Engineering: Principles and Practice*, chap. 30, pp. 1041–1042. UK: Wiley Inc., third ed., 2007.

[28] I. Sommerville. *Software Engineering 9th Edition*, chap. 9, pp. 248–250. USA: Pearson Education Inc., ninth ed., 2011.

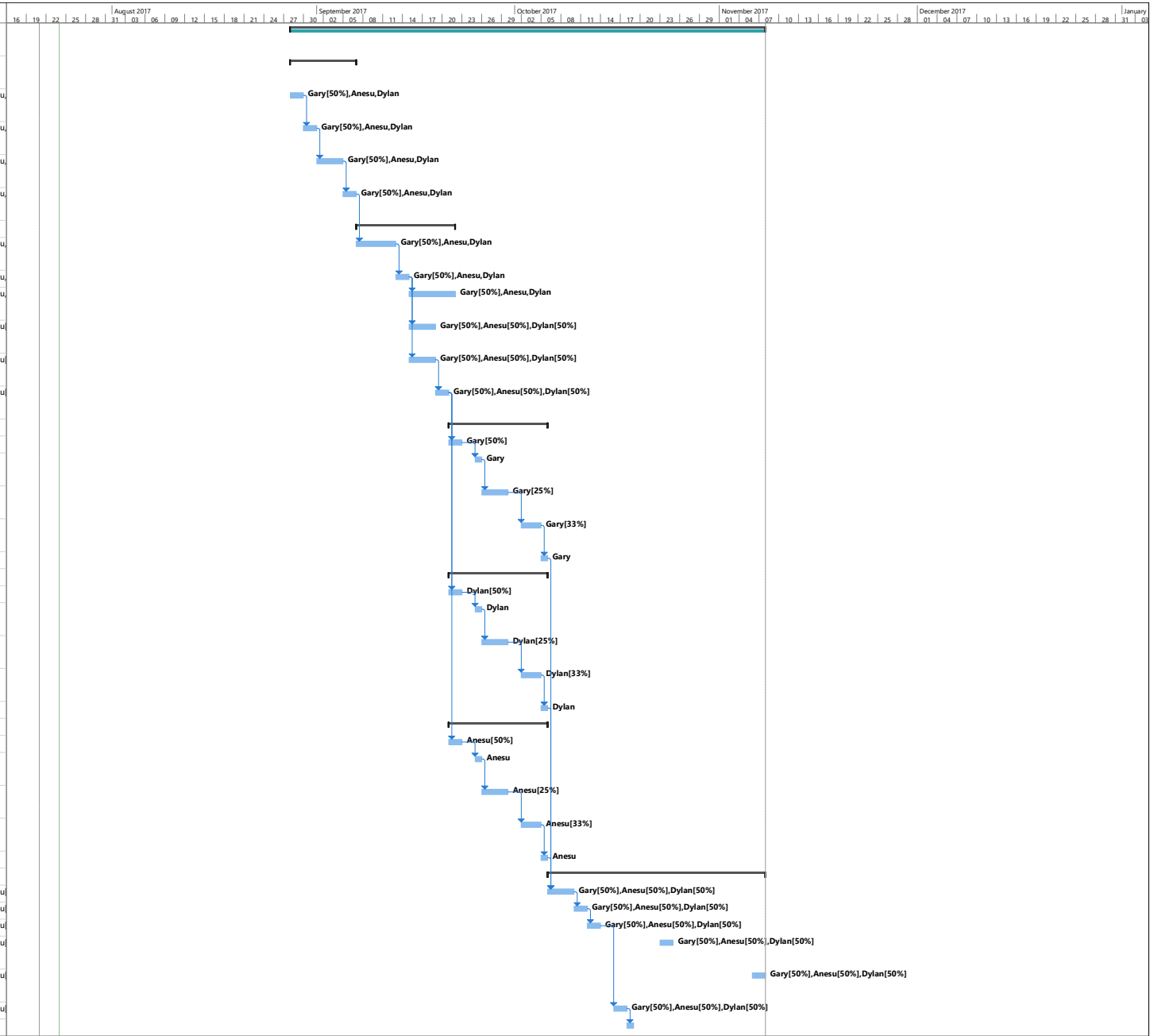
[29] A. Sedlock. "An introduction to front end testing." URL `http://www.creativebloq.com/how-to/an-introduction-to-frontend-testing`.

APPENDIX A

Task Name	Duration	Start	Finish	Resource
GENESIS 2.0	**51 days**	**Mon 28/08/17**	**Mon 06/11/17**	
PRE PLANNING	**8 days**	**Mon 28/08/17**	**Wed 06/09/17**	
Final Design Decisions	2 days	Mon 28/08/17	Tue 29/08/17	All
System Technologies Finalised	2 days	Wed 30/08/17	Thu 31/08/17	All
Development Environment Setup	2 days	Fri 01/09/17	Mon 04/09/17	All
Code Repository Setup	2 days	Tue 05/09/17	Wed 06/09/17	All
SYSTEM FRAMEWORK	**11 days**	**Thu 07/09/17**	**Thu 21/09/17**	
General User Interface	4 days	Thu 07/09/17	Tue 12/09/17	All
Data Input	2 days	Wed 13/09/17	Thu 14/09/17	All
Visualisation Data Models	5 days	Fri 15/09/17	Thu 21/09/17	All
Definition of System Views	2 days	Fri 15/09/17	Mon 18/09/17	All
Definition of Controller	2 days	Fri 15/09/17	Mon 18/09/17	All
PDF Generation Functionality	2 days	Tue 19/09/17	Wed 20/09/17	All
PCA	**11 days**	**Thu 21/09/17**	**Thu 05/10/17**	
Data manipulation	2 days	Thu 21/09/17	Fri 22/09/17	Gary
Visualisation Input Data	1 day	Mon 25/09/17	Mon 25/09/17	Gary
Visualisation Implementation	4 days	Tue 26/09/17	Fri 29/09/17	Gary
Visualisation Interactivity	3 days	Mon 02/10/17	Wed 04/10/17	Gary
Testing	1 day	Thu 05/10/17	Thu 05/10/17	Gary
ADMIXTURE	**11 days**	**Thu 21/09/17**	**Thu 05/10/17**	
Data manipulation	2 days	Thu 21/09/17	Fri 22/09/17	Dylan
Visualisation Input Data	1 day	Mon 25/09/17	Mon 25/09/17	Dylan
Visualisation Implementation	4 days	Tue 26/09/17	Fri 29/09/17	Dylan
Visualisation Interactivity	3 days	Mon 02/10/17	Wed 04/10/17	Dylan
Testing	1 day	Thu 05/10/17	Thu 05/10/17	Dylan
MANHATTAN	**11 days**	**Thu 21/09/17**	**Thu 05/10/17**	
Data Manipulation	2 days	Thu 21/09/17	Fri 22/09/17	Anesu
Visualisation Input Data	1 day	Mon 25/09/17	Mon 25/09/17	Anesu
Visualisation Implementation	4 days	Tue 26/09/17	Fri 29/09/17	Anesu
Visualisation Interactivity	3 days	Mon 02/10/17	Wed 04/10/17	Anesu
Testing	1 day	Thu 05/10/17	Thu 05/10/17	Anesu
GENERAL	**23 days**	**Fri 06/10/17**	**Tue 07/11/17**	
System Integration	2 days	Fri 06/10/17	Mon 09/10/17	All
System Testing	2 days	Tue 10/10/17	Wed 11/10/17	All
Additional Features	2 days	Thu 12/10/17	Fri 13/10/17	All
Open Day Preparation	2 days	Mon 23/10/17	Tue 24/10/17	All
Group Presentation Preparation	2 days	Mon 06/11/17	Tue 07/11/17	All
Documentation	2 days	Mon 16/10/17	Tue 17/10/17	All
Project Wrap Up	1 day	Wed 18/10/17	Wed 18/10/17	All

Table 1: Project Schedule and Work Division

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		**GENESIS 2.0**	**52 days**	**Mon 28/08/17**	**Tue 07/11/17**		
2		**PRE PLANNING**	**8 days**	**Mon 28/08/17**	**Wed 06/09/17**		
3		Final Design Decisions	2 days	Mon 28/08/17	Tue 29/08/17		Gary[50%],Anesu,
4		System Technologies Finalised	2 days	Wed 30/08/17	Thu 31/08/17	3	Gary[50%],Anesu,
5		Development Environment Setup	2 days	Fri 01/09/17	Mon 04/09/17	4	Gary[50%],Anesu,
6		Code Repository Setup	2 days	Tue 05/09/17	Wed 06/09/17	5	Gary[50%],Anesu,
7		**SYSTEM FRAMEWORK**	**11 days**	**Thu 07/09/17**	**Thu 21/09/17**		
8		General User Interface	4 days	Thu 07/09/17	Tue 12/09/17	6	Gary[50%],Anesu,
9		Data Input	2 days	Wed 13/09/17	Thu 14/09/17	8	Gary[50%],Anesu,
10		Visualisation Data Models	5 days	Fri 15/09/17	Thu 21/09/17	9	Gary[50%],Anesu,
11		Definition of System Views	2 days	Fri 15/09/17	Mon 18/09/17	9	Gary[50%],Anesu
12		Definition of Controller	2 days	Fri 15/09/17	Mon 18/09/17	9	Gary[50%],Anesu
13		PDF Generation Functionality	2 days	Tue 19/09/17	Wed 20/09/17	12	Gary[50%],Anesu
14		**PCA**	**11 days**	**Thu 21/09/17**	**Thu 05/10/17**		
15		Data manipulation	2 days	Thu 21/09/17	Fri 22/09/17	13	Gary[50%]
16		Visualisation Input Data	1 day	Mon 25/09/17	Mon 25/09/17	15	Gary
17		Visualisation Implementation	4 days	Tue 26/09/17	Fri 29/09/17	16	Gary[25%]
18		Visualisation Interactivity	3 days	Mon 02/10/17	Wed 04/10/17	17	Gary[33%]
19		Testing	1 day	Thu 05/10/17	Thu 05/10/17	18	Gary
20		**ADMIXTURE**	**11 days**	**Thu 21/09/17**	**Thu 05/10/17**		
21		Data manipulation	2 days	Thu 21/09/17	Fri 22/09/17	13	Dylan[50%]
22		Visualisation Input Data	1 day	Mon 25/09/17	Mon 25/09/17	21	Dylan
23		Visualisation Implementation	4 days	Tue 26/09/17	Fri 29/09/17	22	Dylan[25%]
24		Visualisation Interactivity	3 days	Mon 02/10/17	Wed 04/10/17	23	Dylan[33%]
25		Testing	1 day	Thu 05/10/17	Thu 05/10/17	24	Dylan
26		**MANHATTAN**	**11 days**	**Thu 21/09/17**	**Thu 05/10/17**		
27		Data Manipulation	2 days	Thu 21/09/17	Fri 22/09/17	13	Anesu[50%]
28		Visualisation Input Data	1 day	Mon 25/09/17	Mon 25/09/17	27	Anesu
29		Visualisation Implementation	4 days	Tue 26/09/17	Fri 29/09/17	28	Anesu[25%]
30		Visualisation Interactivity	3 days	Mon 02/10/17	Wed 04/10/17	29	Anesu[33%]
31		Testing	1 day	Thu 05/10/17	Thu 05/10/17	30	Anesu
32		**GENERAL**	**23 days**	**Fri 06/10/17**	**Tue 07/11/17**		
33		System Integration	2 days	Fri 06/10/17	Mon 09/10/17	19,25,31	Gary[50%],Anesu
34		System Testing	2 days	Tue 10/10/17	Wed 11/10/17	33	Gary[50%],Anesu
35		Additional Features	2 days	Thu 12/10/17	Fri 13/10/17	34	Gary[50%],Anesu
36		Open Day Preparation	2 days	Mon 23/10/17	Tue 24/10/17		Gary[50%],Anesu
37		Group Presentation Preparation	2 days	Mon 06/11/17	Tue 07/11/17		Gary[50%],Anesu
38		Documentation	2 days	Mon 16/10/17	Tue 17/10/17	35	Gary[50%],Anesu
39		Project Wrap Up	1 day	Wed 18/10/17	Wed 18/10/17	38	



Project: Lab Project Project Sch
Date: Mon 24/07/17

Task | Split | Milestone | Summary | Project Summary | Inactive Task | Inactive Milestone | Inactive Summary | Manual Task | Duration-only | Manual Summary Rollup | Manual Summary | Start-only | Finish-only | External Tasks | External Milestone | Deadline | Progress | Manual Progress