DEVELOPMENT OF A COMPUTATIONAL TOOL FOR FORENSIC DNA ANALYSIS

By

ANURAG ARNOLD

A thesis submitted to the

Graduate School-Camden

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of Master of Science

Graduate Program in Computer Science

Written under the direction of

Dr. Desmond S. Lun

And approved by

Dr. Desmond S. Lun

Dr. Suneeta Ramaswami

Dr. Dawei Hong

Camden, New Jersey

January 2016

THESIS ABSTRACT

Development of a Computational Tool for Forensic DNA Analysis

By ANURAG ARNOLD

Thesis Director:
Dr. Desmond S. Lun

Forensic DNA analysis consists of a DNA profiling process by a method called as STR Analysis, short for Short Tandem Repeats. By using the statistics it provides, various probabilistic approaches are implemented in a software package to find out DNA profile matches and individual identifications. This study has contributed in the design, development and integration of an easy to use and interactive software application to find the number of contributors in a DNA sample and to find a match against it with another person's DNA. This research specifically deals with the (1) design of an algorithm for filtering unfiltered DNA profile samples to remove bleed-through peaks; (2) design of an interactive interface for MatchIt, where a DNA sample of a person of interest is tested against a DNA sample to find a match; and (3) data modeling, design and integration of a database for the software.

ACKNOWLEDGEMENTS

I would like to express my most sincere gratitude and appreciation to Dr. Desmond Lun, my thesis advisor and mentor, for his patience, guidance and encouragement all along. My grateful thanks to him for giving me an opportunity to work in an excellent atmosphere to work on my research. I would not have been able reach this point without all the support from my advisor, committee members, colleagues, friends and family.

My sincere thanks to Dr. Sunil Shende for his guidance and helping me in understanding the fundamentals of computer science and big data algorithms.

My sincere thanks to Dr. Dawei Hong, Dr. Michael Palis and Dr. J C Birget who helped me to develop my background in optimization methods, parallel computing and numerical methods.

I would like to thank Dennis Egen who has been a great mentor to me and helped me in understanding software engineering.

I would like to thank Harish Swaminathan, Abhishek Garg and Lauren E. Alfonse for all their help and suggestions. My research would not have been possible without their help.

I would like to express my special thanks to Dr. Suneeta Ramaswami for giving me an opportunity to be a part of Department of Computer Science at Rutgers Camden, and for her esteemed guidance.

Finally, I would like to thank my family for their support and encouragement throughout my study.

TABLE OF CONTENTS

Introduction

Forensic Science has been used extensively for criminal investigation purposes for past three decades. The recent developments in the field utilizes a technique known as Short Tandem Repeat Analysis. The STR is a repetitive DNA sequence in the form of 2-5 base pairs of nitrogenous bases found all through the DNA present in living organisms. These nitrogenous bases consist of hydrogen bonds with each other connecting the two anti-parallel strands that form the helix-shaped DNA.

STR analysis used for the purpose of DNA profiling is based on PCR (Polymerase Chain Reaction). PCR amplifies a DNA template consisting of the STR sequences which is then passed through capillary electrophoresis to produce a DNA profile of an individual. The highly variable repetitive sequences used in DNA profiling called as Variable number tandem repeats are extremely unlikely to be the same for two individuals. This helps in identifying and matching a DNA sample against an individual.

The DNA profile, an electropherogram, is used to determine the number of contributors present in the sample after the preprocessing.

Based on the assumption of number of contributors, a commonly used approach known as Likelihood Ratio is used to calculate the match statistic.

The Likelihood Ratio is defined as:

$$LR = \frac{Pr\left(E \middle| H_p, n_p\right)}{Pr(E|H_d, n_d)},$$

where, E is the evidence in the form of the electropherogram (epg); H_p and H_d are the hypotheses specified by the prosecution and the defense, respectively; and n_p and n_d are the number of contributors specified by the prosecution and the defense, respectively. The numerator is the probability of observing the evidence given the prosecution's hypothesis and the denominator is the probability of observing the evidence given the defense's hypothesis. The evidence shows support for the prosecution's hypotheses if LR > 1; if LR < 1 the defense's hypothesis is supported by the evidence. The calculation of a Likelihood Ratio depends upon an assumption about the number of contributors both in the numerator as well as the denominator, making it essential to have a good estimate about the number of contributors to accurately calculate a statistic that represents the information captured in the signal. Thus, utilizing a number of contributors that is not representative of the actual number that gave rise to a sample may affect the interpretation of the sample's profile.
The p-value for the suspect is defined as the probability that a randomly picked person from the population would give rise to an LR at least as large as the one observed for the suspect.

$$p - value(s) = \left(LR(R) \geq LR(s)\right)$$

Using the above approach to find the number of contributors in a DNA sample and match it against that of a person of interest, we need an efficient and interactive software solution that works with the underlying algorithms on the data, and assists in generating and visualizing test results. This project contributes in the development of a concise solution for the challenges faced in designing a software for DNA analysis.

Purpose

The purpose of my thesis project has been to contribute in the design, development, and integration of the various components of a computational software tool for Forensic DNA Analysis. The first part of my thesis consists of an implementation of a Filtering Algorithm to filter testing samples to remove the bleed-through and extraneous peaks of alleles resulting from misreading during electrophoresis detection. The testing samples are used for generating a calibration file that is used in the algorithms used for the finding the number of contributors in a DNA sample and matching it against a person of interest's DNA sample. The part of the application that refers to finding the number of contributors is called as NOCit. And the third part of the software is called MatchIt that tries to find a match between a DNA sample and a person of interest. The bleed-through peaks of alleles need to be removed from the DNA data before being analyzed in NOCit and MatchIt.

Next, I have designed a user interface for the MatchIt algorithm, which matches a suspect's DNA sample against the one found at a crime scene. This interface has been integrated with the rest of the software. Thirdly, I have designed a schema for a relational database and integrated it with the various components of the software to assist with the data being used.

My work can be summarized as,

1. Designing and implementing a Filtering Algorithm.
2. Developing a simple and user-friendly user interface for MatchIt.
3. Data modelling, designing and integrating a Relational Database with the software.

Design and Implementation of Filtering Algorithm

3.1 Introduction

DNA profiling for forensic purposes starts with a sample of the DNA being amplified at specific STR loci using Polymerase Chain Reaction (PCR). The next step is to pass the resulting DNA fragments through an electrophoresis process. This is the preprocessing stage wherein the DNA profile of the person is generated. The data present in the profile is then used for further analysis. Before the analysis, however, there can be some noise that is present in the profile. These are in the form of spurious peaks in the electropherogram, and they need to be filtered out.

3.2 Theory

After the DNA has been collected, we use Polymerase Chain Reaction to amplify specific STR loci. The amplification results in copies of the DNA at the loci so that they can then be detected during electrophoresis. The fragments of DNA produced are separated and detected by capillary electrophoresis technique. During this process, fluorescent tags attached to the DNA emit light in the visible spectrum when they are passed through the capillary in UV light.

The PCR technique is carried out using the Identifiler Plus kit that produces fifteen known set of loci with their set of alleles present in them. During electrophoresis, the detector detects the color emitted by the fragments of DNA and categorizes the alleles for each loci with their size and height accordingly. A screenshot of a sample file produced is shown in Figure 3.1.

Sometimes due to the misreading of the detector, there may be false allele peaks present in a particular dye color. There are several ways in which the peaks appear in a dye color channel they do not belong to. For example, the detector for the color green might also detect some of the portions emitting at the adjacent color, yellow, in the spectrum. These extraneous peaks are known as the bleed-through peaks. Before analyzing and processing the DNA data, it is important to remove all these bleed-through peaks. A three-step process implemented in the filtering algorithm for this has been described in the next section.

3.3 Filtering Algorithm

An unfiltered sample file consists of the size and height of all the alleles at a specific dye color channel present at each locus, along with more data from the preprocessing such as the run name and the amount of the DNA sample. The filtering algorithm is designed for the unfiltered sample files with bleed-through peaks to get filtered for the calibration data to be generated, and also for filtering the NOCit and MatchIt testing sample files. It is basically comprised of a three-step process, and the algorithm has been integrated with the user interface as well.

3.3.1 Color of Bleed-Through Peak

This is the first step in identifying the bleed-through peaks. The first restriction is that the dye color of the bleed-through peak cannot be the same as that of the true peak as the bleed-through appears in a color channel they are not supposed to. We check all the pairs of alleles having different dye colors. The algorithm uses a Java Collections Hashmap to arrange the peaks and checks and considers only the ones that are different in color from the other peaks. So once this criteria is satisfied, the second criteria is considered.

3.3.2 Size of Bleed-Through Peak

The second criteria checks the pairs of different colored peaks for the absolute value of their size difference. The size represents the distance between two base pairs present in the STR loci. If the difference of sizes of the two alleles is less than or equal to a threshold of 0.3, the criteria gets satisfied and the third criteria will be checked for the peaks. The threshold of 0.3 can be changed by the user in the software through an interface.

3.3.3 Height of Bleed-Through Peak

The third constraint is to check that if the ratio of the peak heights of the two allele pairs are less than 5%. If this check is passed, then the first peak gets removed as the bleed-through. The same procedure is carried out for all the pairs of the alleles present in different dye colors.

3.4 Algorithm Pseudocode

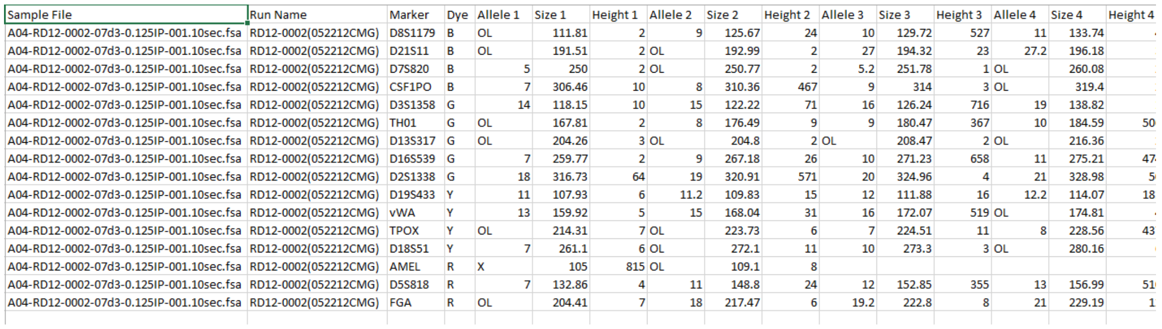
The algorithm takes in a HashTable as input, hashed with the colors, Red, Blue, Green and Yellow, containing a List of (Size, Height) pairs for each color.

For every peak in one color, we have to find if there are any peaks in other colors whose difference in size and height ratio is less than the threshold. If there is a peak whose height is less than a corresponding peak's height and their size difference less than the size threshold, then this peak is removed.

```
Input: HashTable, H
H[Color] is a List of (Size, Height) pairs,
for colors: 'R', 'G', 'B' and 'Y'


IF ∃(Size1, Height1) pair in H[Color1]
     AND (Size2, Height2) pair in H[Color2]
WHERE, |Size1 - Size2| < Size Threshold
     AND Height1 < Height2 * Height Threshold
THEN, REMOVE (Size1, Height1)


Output: HashTable, H
H[Color] is a List of (Size, Height) pairs, with the peaks
removed.
```

3.5 Testing and Results

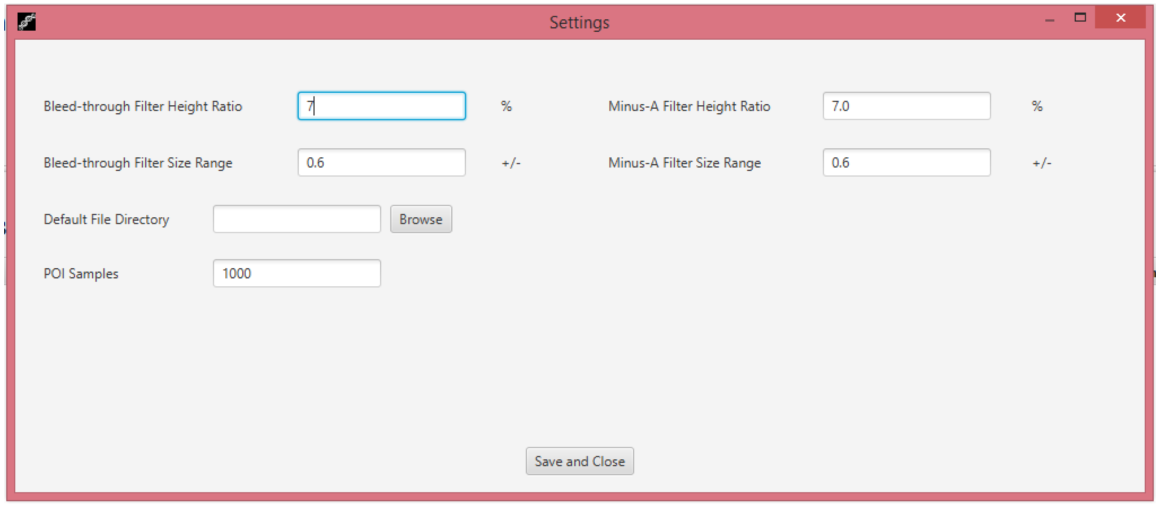
During the calibration process, the application has checkboxes for the user to select whether the samples being used for calibration need to be filtered or not. The following snapshot in Figure 3.1 shows a snippet of the sample data file with the peaks. The letters B, G, Y and R represent the dye colors in the Dye columns, referring to Blue, Green, Yellow and Red respectively.



Sample File	Run Name	Marker	Dye	Allele 1	Size 1	Height 1	Allele 2	Size 2	Height 2	Allele 3	Size 3	Height 3	Allele 4	Size 4	Height 4
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	D8S1179	B	OL	111.81	2	9	125.67	24	10	129.72	527	11	133.74	
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	D21S11	B	OL	191.51	2	OL	192.99	2	27	194.32	23	27.2	196.18	
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	D7S820	B	5	250	2	OL	250.77	2	5.2	251.78	1	OL	260.08	
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	CSF1PO	B	7	306.46	10	8	310.36	467	9	314	3	OL	319.4	
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	D3S1358	G	14	118.15	10	15	122.22	71	16	126.24	716	19	138.82	
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	TH01	G	OL	167.81	2	8	176.49	9	9	180.47	367	10	184.59	50
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	D13S317	G	OL	204.26	3	OL	204.8	2	OL	208.47	2	OL	216.36	
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	D16S539	G	7	259.77	2	9	267.18	26	10	271.23	658	11	275.21	47
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	D2S1338	G	18	316.73	64	19	320.91	571	20	324.96	4	21	328.98	5
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	D19S433	Y	11	107.93	6	11.2	109.83	15	12	111.88	16	12.2	114.07	18
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	vWA	Y	13	159.92	5	15	168.04	31	16	172.07	519	OL	174.81	
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	TPOX	Y	OL	214.31	7	OL	223.73	6	7	224.51	11	8	228.56	43
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	D18S51	Y	7	261.1	6	OL	272.1	11	10	273.3	3	OL	280.16	
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	AMEL	R	X	105	815	OL	109.1	8						
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	D5S818	R	7	132.86	4	11	148.8	24	12	152.85	355	13	156.99	51
A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	RD12-0002(052212CMG)	FGA	R	OL	204.41	7	18	217.47	6	19.2	222.8	8	21	229.19	1

Figure 3.1: Screenshot of a sample file.

Also, with a Menu option in Settings, shown in Figure 3.2, the user can set the thresholds for the size difference and the height ratios to be used for filtering the peaks in the step 2 and 3.



Bleed-through Filter Height Ratio: 7 %

Bleed-through Filter Size Range: 0.6 +/-

Minus-A Filter Height Ratio: 7.0 %

Minus-A Filter Size Range: 0.6 +/-

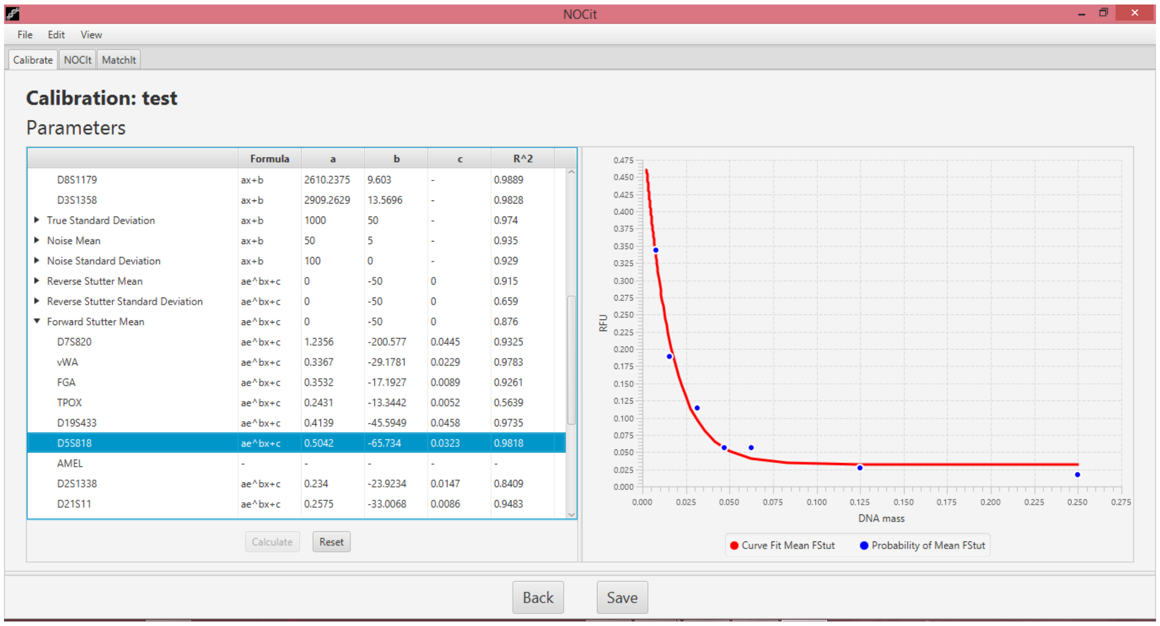
Default File Directory:

POI Samples: 1000

Figure 3.2 Interface for setting the thresholds for the size difference and peak height ratio of alleles.

Unfiltered testing samples were filtered and compared to the ones not unfiltered in regards to the Calibration data generated, and on the NOCit results.

Figures 3.3 and Figure 3.4 show a comparison of the results between a Calibration data of unfiltered samples and filtered samples.

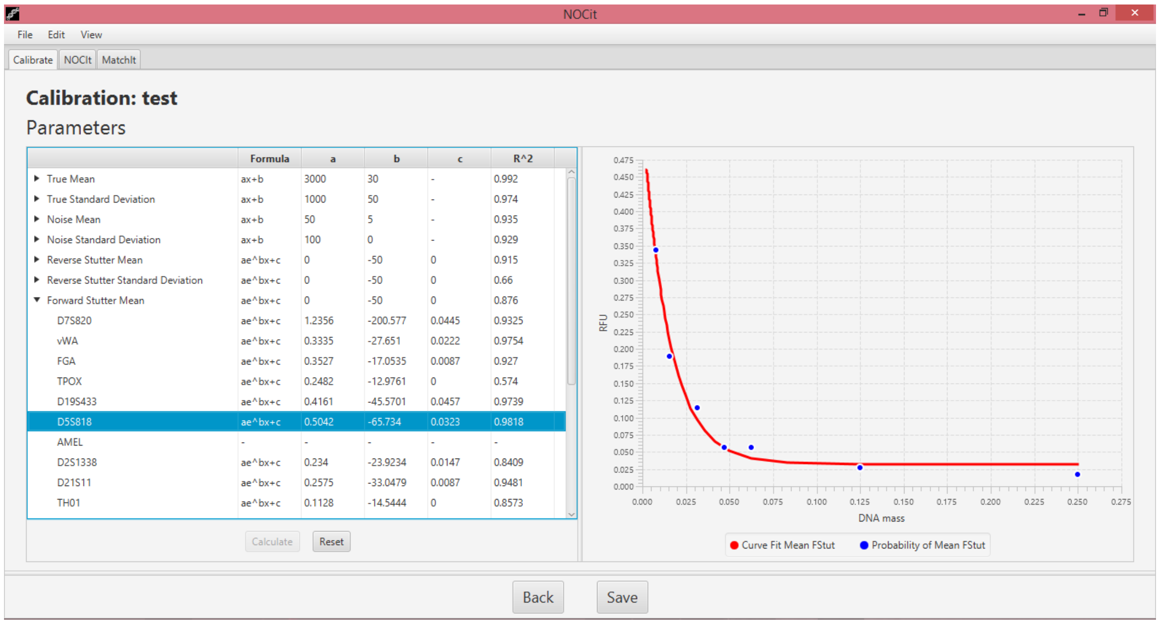


Calibration: test

Parameters

	Formula	a	b	c	R^2
D8S1179	ax+b	2610.2375	9.603	-	0.9889
D3S1358	ax+b	2909.2629	13.5696	-	0.9828
▸ True Standard Deviation	ax+b	1000	50	-	0.974
▸ Noise Mean	ax+b	50	5	-	0.935
▸ Noise Standard Deviation	ax+b	100	0	-	0.929
▸ Reverse Stutter Mean	ae^bx+c	0	-50	0	0.915
▸ Reverse Stutter Standard Deviation	ae^bx+c	0	-50	0	0.659
▾ Forward Stutter Mean	ae^bx+c	0	-50	0	0.876
D7S820	ae^bx+c	1.2356	-200.577	0.0445	0.9325
vWA	ae^bx+c	0.3367	-29.1781	0.0229	0.9783
FGA	ae^bx+c	0.3532	-17.1927	0.0089	0.9261
TPOX	ae^bx+c	0.2431	-13.3442	0.0052	0.5639
D19S433	ae^bx+c	0.4139	-45.5949	0.0458	0.9735
D5S818	ae^bx+c	0.5042	-65.734	0.0323	0.9818
AMEL	-	-	-	-	-
D2S1338	ae^bx+c	0.234	-23.9234	0.0147	0.8409
D21S11	ae^bx+c	0.2575	-33.0068	0.0086	0.9483

Figure 3.3 Sample graph for a locus from unfiltered sample files.



Calibration: test

Parameters

	Formula	a	b	c	R^2
True Mean	ax+b	3000	30	-	0.992
True Standard Deviation	ax+b	1000	50	-	0.974
Noise Mean	ax+b	50	5	-	0.935
Noise Standard Deviation	ax+b	100	0	-	0.929
Reverse Stutter Mean	ae^bx+c	0	-50	0	0.915
Reverse Stutter Standard Deviation	ae^bx+c	0	-50	0	0.66
Forward Stutter Mean	ae^bx+c	0	-50	0	0.876
D7S820	ae^bx+c	1.2356	-200.577	0.0445	0.9325
vWA	ae^bx+c	0.3335	-27.651	0.0222	0.9754
FGA	ae^bx+c	0.3527	-17.0535	0.0087	0.927
TPOX	ae^bx+c	0.2482	-12.9761	0	0.574
D19S433	ae^bx+c	0.4161	-45.5701	0.0457	0.9739
D5S818	ae^bx+c	0.5042	-65.734	0.0323	0.9818
AMEL	-	-	-	-	-
D2S1338	ae^bx+c	0.234	-23.9234	0.0147	0.8409
D21S11	ae^bx+c	0.2575	-33.0479	0.0087	0.9481
TH01	ae^bx+c	0.1128	-14.5444	0	0.8573

Figure 3.4 Sample graph for a locus from the filtered sample files.

Development of a Graphical User Interface for MatchIt

4.1 Introduction

As important as it is to have an efficient algorithm, it is equally important to have a simple user interface and experience in using the software. In designing the MatchIt interface for our software tool, it was important to make the features and flow of work similar to the other parts of the software. Also important was to make sure the user experience is smooth and interactive at every step, for example during the time the calculations are being carried out and at the end, the results that need to be show with a proper graph.

4.2 Design Pattern

The design pattern used in the project is a layer approach called MVC, short for Model-View-Controller. The MVC is an architectural pattern used in the design of User Interfaces. The code and the structure is organized in such a way that the Controller sends commands to the Data Model layer and also to the View Layer for updating any events triggered in the interface by the user like accepting input data, with the click of a button. The Model layer contains all the algorithmic data coupled with the database backend. It has all the behavioral aspects of the application and is independent of the user interface. The View layer has all the UI components like the style sheets and FXML files that hold the whole structure of the application. It contains the views for all the three interactive sections of the application including the graphs generated.

Figure 4.1 shows the screenshot of the code organization with the help of a class diagram.

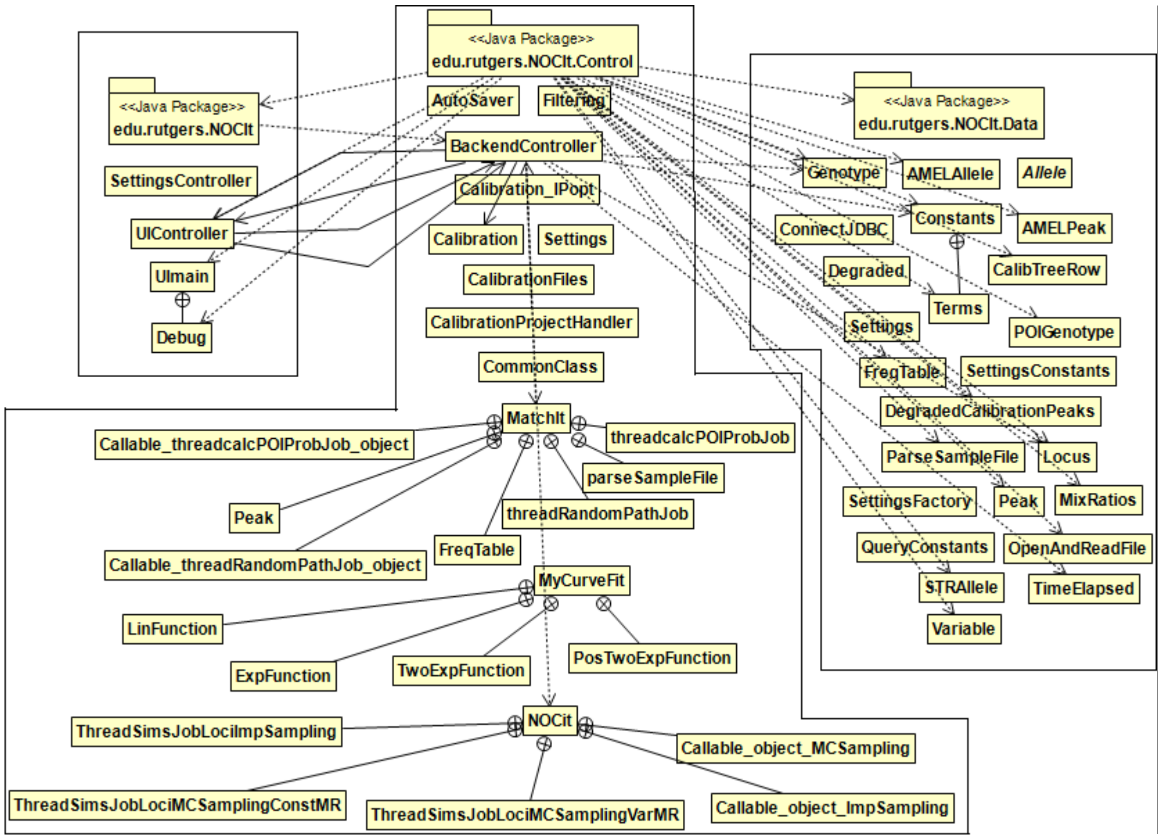


Figure 4.1: Class diagram showing all the class relationships in the MVC design pattern.

The first package consisting of the UI Controller classes has all the UI components, and the Control package contains the backend controller logic that accepts input and sends data to the Model-Database, which is in the third package, Data.

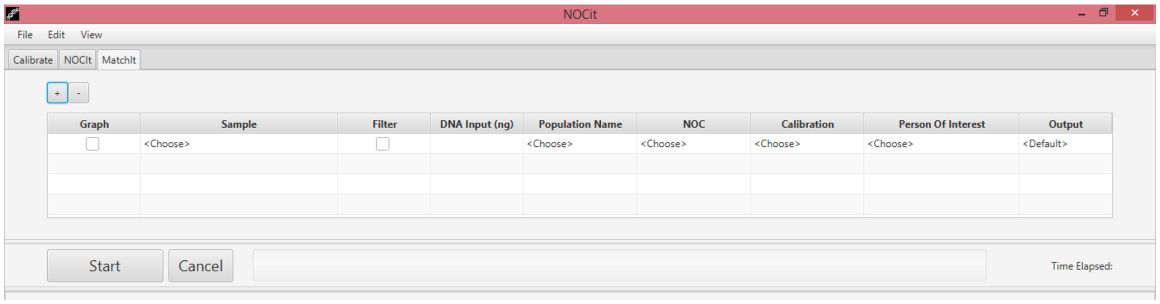
4.3 Simple and Friendly User Interface

As the software workflow is divided into three sections, namely Calibration of data, NOCit for finding out the number of contributors in a DNA sample, and MatchIt for matching the DNA sample with the suspect's DNA. I have worked on designing and developing the MatchIt interface with a similar user experience as in NOCit. This includes a section for input data with validation and integrity checks against the database, calculations and results display with a histogram plot.

4.3.1 Input Calibration Data and Person of Interest

The Input section has nine columns for Graph display checkbox, Sample file chooser, Filter checkbox for filtering the sample file, DNA Input in nanograms, Population Name dropdown selection, NOC, Calibration files saved earlier, Person of Interest genotype sample chooser, and Output file destination selection for the pdf report that gets generated.

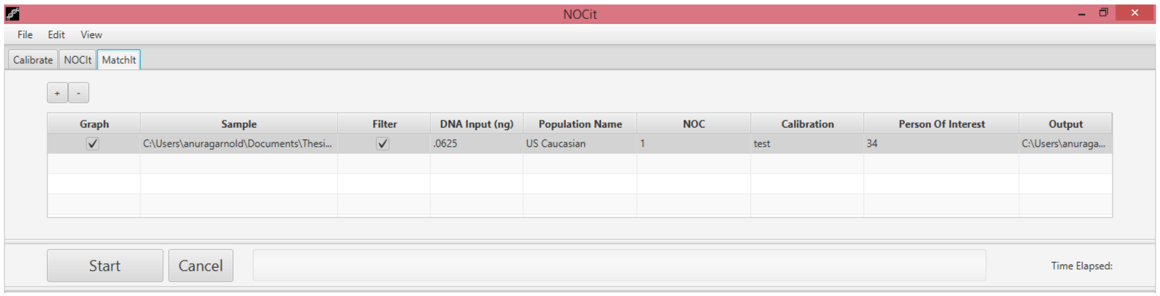
Figure 4.2 shows a screenshot of the input section of the interface with no input data yet.



Graph	Sample	Filter	DNA Input (ng)	Population Name	NOC	Calibration	Person Of Interest	Output
☐	<Choose>	☐		<Choose>	<Choose>	<Choose>	<Choose>	<Default>

Figure 4.2: MatchIt interface with a table for all the inputs.

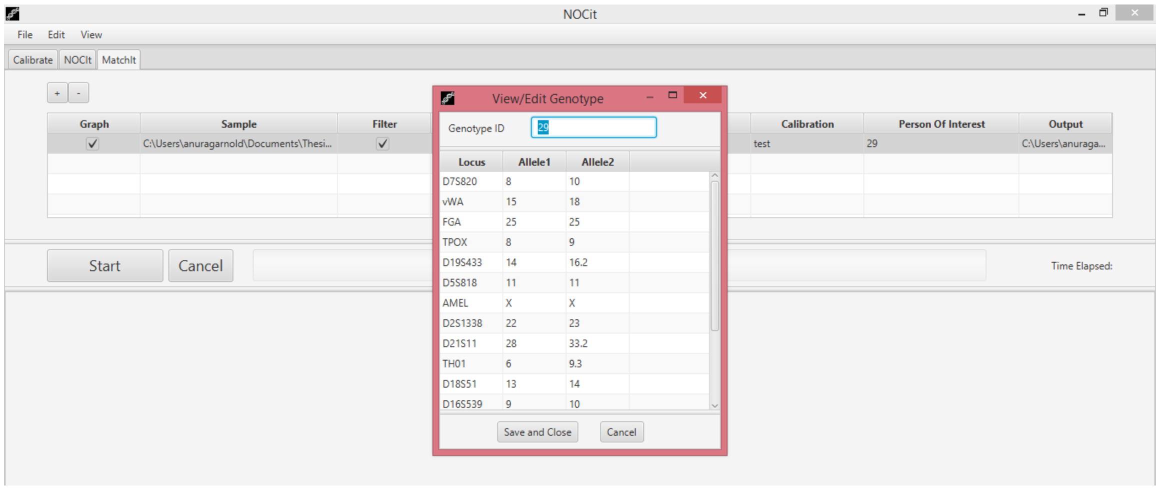
Figure 4.3 shows the same section when all input data has been accepted and is ready to be validated



Graph	Sample	Filter	DNA Input (ng)	Population Name	NOC	Calibration	Person Of Interest	Output
✓	C:\Users\anuragarnold\Documents\Thesi...	✓	.0625	US Caucasian	1	test	34	C:\Users\anuraga...

Figure 4.3: MatchIt interface with all the input parameters.

Also added is the context menu options that are consistent with the design in the first section of the software where Calibration Genotypes are displayed. Figure 4.4 shows the screenshot.



Graph	Sample	Filter	Calibration	Person Of Interest	Output
✓	C:\Users\anuragarnold\Documents\Thesi...	✓	test	29	C:\Users\anuraga...

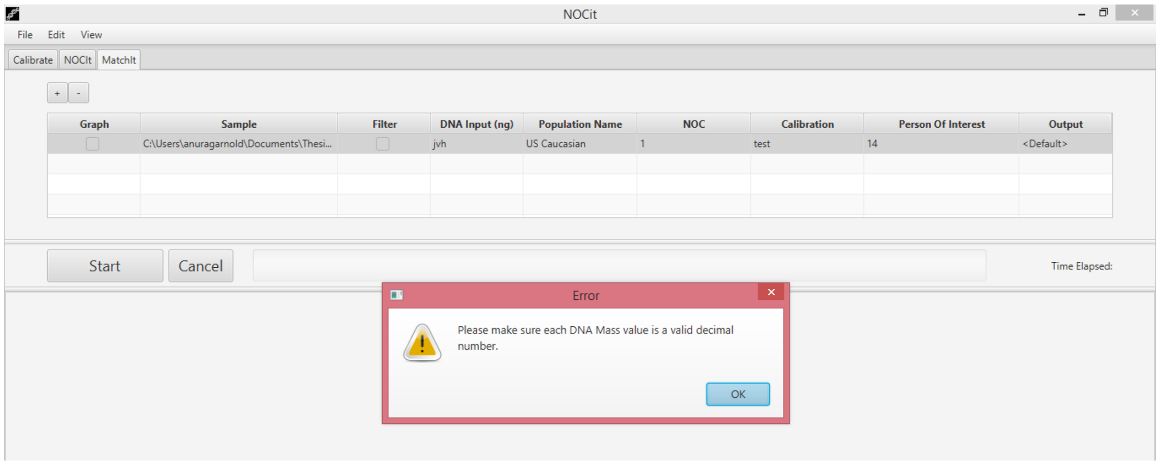
Genotype ID: 29

Locus	Allele1	Allele2
D7S820	8	10
vWA	15	18
FGA	25	25
TPOX	8	9
D19S433	14	16.2
D5S818	11	11
AMEL	X	X
D2S1338	22	23
D21S11	28	33.2
TH01	6	9.3
D18S51	13	14
D16S539	9	10

Figure 4.4: Pop-up interface for viewing and editing the genotype file for the person of interest.

After the input data is accepted it is validated against two conditions, first whether all the required data is present in the specified format and against the database for the loci, allele and frequency values for the selected population type.

Figure 4.5 shows a screenshot that checks for a valid DNA input. For the validation checks against database, see section 5.3.3.



Graph	Sample	Filter	DNA Input (ng)	Population Name	NOC	Calibration	Person Of Interest	Output
☐	C:\Users\anuragarnold\Documents\Thesi...	☐	jvh	US Caucasian	1	test	14	<Default>

Please make sure each DNA Mass value is a valid decimal number.

Figure 4.5: Validation check warning for a valid DNA input.

4.3.2 Running the Test

Once all the data has been validated and input files uploaded, the MatchIt algorithm and calculations start when the Start button is clicked. There is a progress bar with a timer that signifies the running of the algorithm. The smaller the Number of Contributors, the less time is taken to calculate the results. This step has been added with a multi-threaded approach so that the software functions in parallel and shows the results when the calculations end. The multi-threading has been implemented using the JDK 8 Thread API. Without parallelism, the software running on a single thread would freeze until the calculations complete.

Example of code snippet:

```java
final BackendController backendControllerMit = this;
        Thread matchitThread = new Thread() {
            public synchronized void run() {
                synchronized (uiControllerMit) {
                    try {
                        MatchIt matchit = new MatchIt(calibration);
                        matchit.runMatchIt(outputFile, frequencyFile, sampleFile,
dnaMass, maxNOC, genotypepoi, calculationsProgressBar, poiSamples, kitName, popName);
                        backendControllerMit.MatchItThreadFinished(this,
matchit.graphBarChart(outputFile), matchit.getMatchItResultsString(), outputFile,
count);
                    } catch (final Exception e) {
                        Platform.runLater(new Runnable() {
                            @Override
                            public void run() {
logger.error(Constants.RUN_MATCHIT_THREAD_ERROR_LOG_MESSAGE, e);
                            }
                        });
                    }
                }
            }
        };
        matchitThread.start();
        threadListMit.add(matchitThread);
        this.threadCount++;
```

The above code starts a new Thread by calling the Thread API in Java. It locks the monitor on the uiControllerMit object and calls the runMatchIt function in the MatchIt.java class. This code resides in the BackendController.java class, and once the MatchIt calculations are finished, the backendController object calls the functions that generate the MatchIt Histogram plot along with the printed result output.

From the architectural point of view, as the user clicks on the Start button for the MatchIt, the event generated will be accepted by the UIController that passes all the data to the backend controller, which calls the backend code for the MatchIt data processing. All the constants and algorithms implemented reside in the layer as the MatchIt class. When the results are generated, they are passed back to the backend controller which passes it to the UIController. The UIController sends the data to the respective nodes in the user interface.

4.3.3 Displaying Results and Graph

After the calculations are over, the results get displayed in the bottom and the third pane of the interface, and depending on the Graph display option ticked in the input section, it visualizes the results as a Histogram Plot. The result set includes the sample details that were given as input, with the True Likelihood Ratio for the Person of Interest, Log (POI LR). The histogram plots the frequency values for all the Log (POI LR) values less than the True Likelihood Ratio as yellow in color, and the ones greater than True Likelihood Ratio as Green in color.

A screenshot of the results that is displayed with the histogram in the application is shown in Figure 4.6

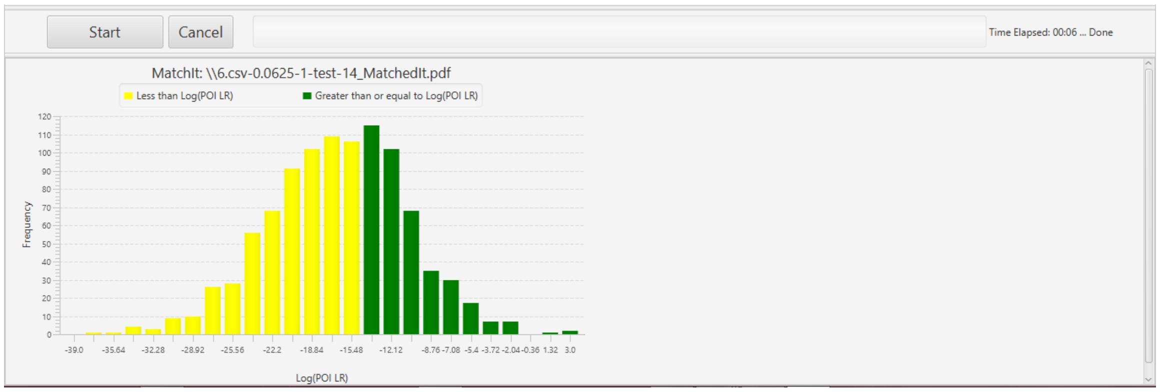
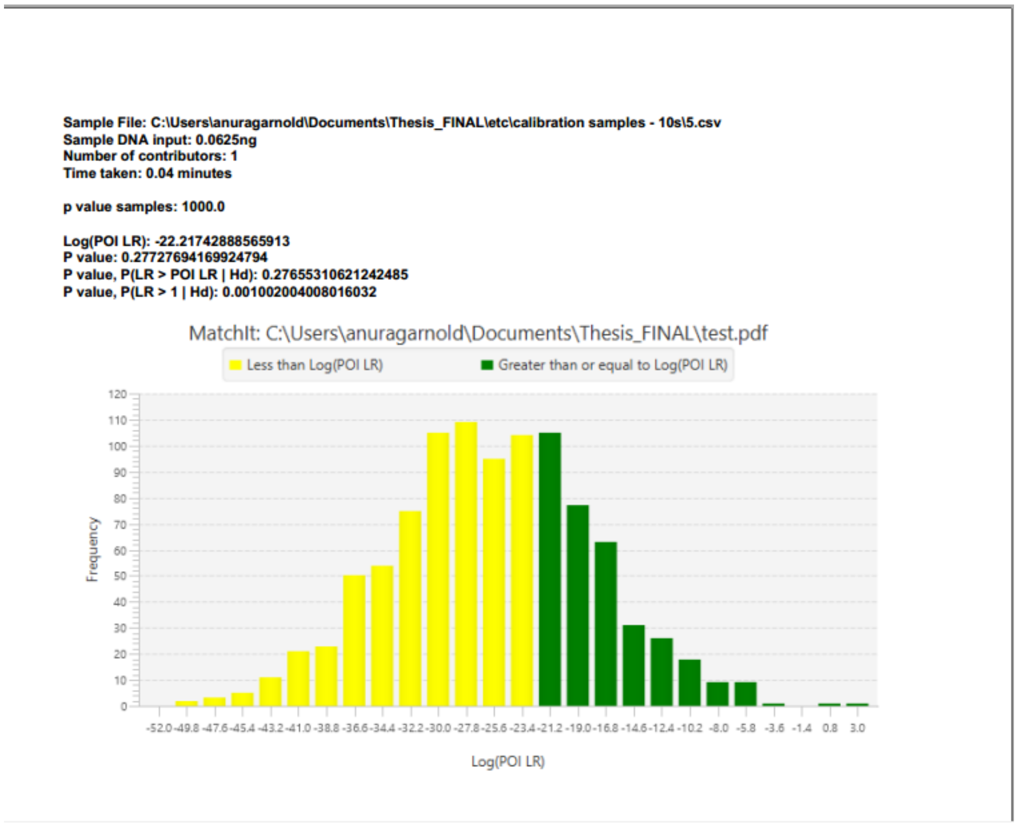


Figure 4.6: MatchIt results shown in the application with a histogram plot.

4.4 Results

A screenshot of the pdf report generated and saved in the user's disk is shown in Figure 4.7



```
Sample File: C:\Users\anuragarnold\Documents\Thesis_FINAL\etc\calibration samples - 10s\5.csv
Sample DNA input: 0.0625ng
Number of contributors: 1
Time taken: 0.04 minutes

p value samples: 1000.0

Log(POI LR): -22.21742888565913
P value: 0.27727694169924794
P value, P(LR > POI LR | Hd): 0.27655310621242485
P value, P(LR > 1 | Hd): 0.001002004008016032
```

Figure 4.7: Screenshot of the pdf generated with the MatchIt results.

Data Modelling and Integration of a Relational Database

5.1 Introduction

Any application with data processing needs to store data persistently and in a structured manner. The data in this application consists of a Kit, and a set of Loci for a specific Kit. Each Locus is composed of a set of Alleles and specifies a location or position of a DNA sequence present on a chromosome. The Kit used here refers to the PCR amplification that produces a set of fifteen known Loci and is known as Identifiler Plus. The Alleles representing a variation of the DNA sequence at each locus, are in turn associated with a specific Population type with their relative Population frequencies. For each different Population type and for a specific Kit, the Alleles will have different frequencies at their respective Locus values. The whole data model can be visualized as consisting of a nested structure with Kit containing Loci containing Alleles, the Alleles also being associated with a specific Population type with their different Frequencies, with the Frequencies adding up to 1.0 for all the Alleles at a Locus.

For storing this data in a database it needed to be first modeled with all the relations, compositions and associations between each of them. A UML diagram in the Section 5.2 shows the data model. The process from Calibration to NOCit and MatchIt in the application uses a lot of this data that needed to be organized, edited and retrieved when needed.

The problem of having all the sample files in their formats for different types of Kits and Populations to be used by the application was solved with the help of a Relational Database. Being a light and easy to use database management system, SQLite was the best choice.

The database consists primarily of the set of Loci belonging to the kind of Kit selected, Alleles present in the Loci with their frequencies for the different Population Types. One of the Kits that is used is called as Identifiler Plus, which consists of a set of fifteen known Loci. Identifiler Plus is a PCR Amplification kit that uses Short Tandem Analysis technique and amplifies the STR Loci and Amelogenin in a single tube. The database schema has been designed and normalized using UML modelling techniques shown in the next section.

5.2 Design and Schema for SQLite Database

The technique used for the data modelling and designing the schema for the database is UML, short for Unified Modelling Language. A UML diagram is a general purpose modelling language to visualize the design of a database system.

The UML diagram consisting of two compositions and associations between the data is shown in Figure 5.1. Kit is composed of Locus and Locus in turn is composed of Allele. The Allele has an association with the Population related by its Frequency.

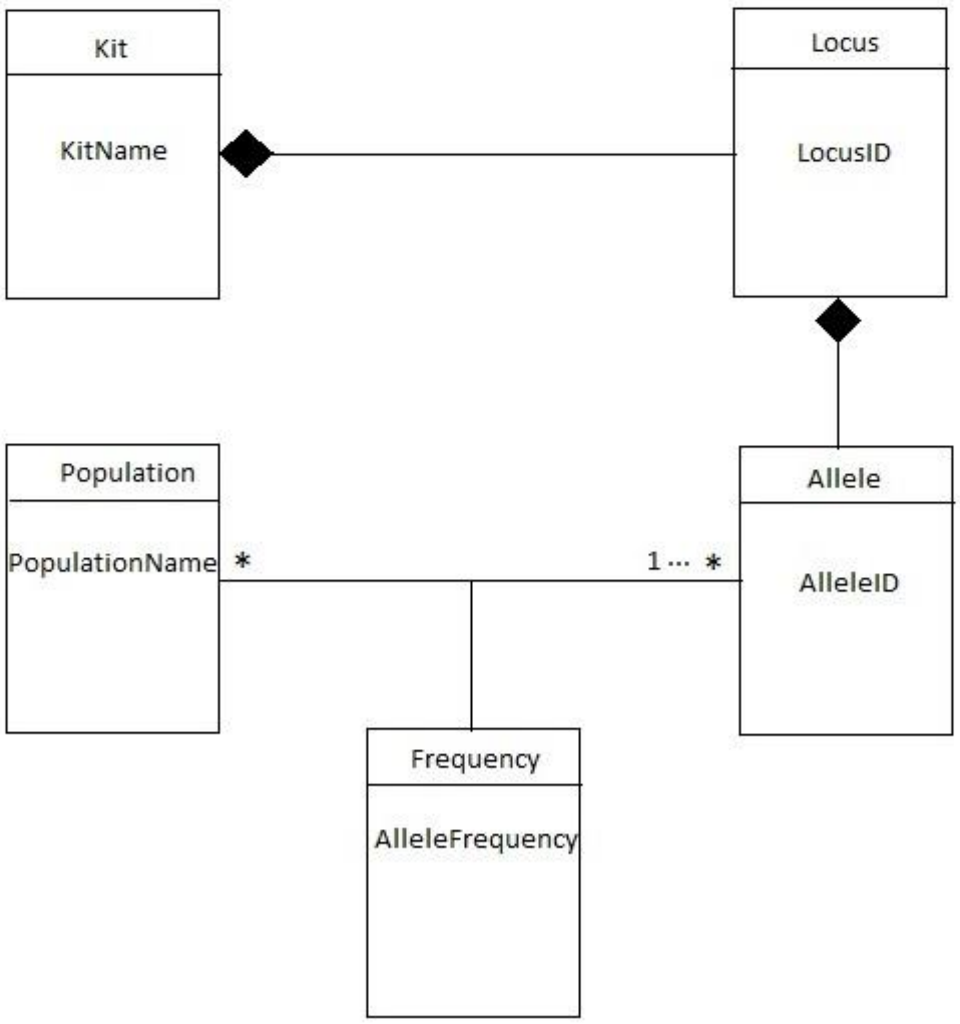


Figure 5.1: UML diagram for the database schema design.

The UML data models were transformed into relations by breaking down the compositions and the associations within it. After normalization, the relational schema designed consisted of five tables namely,

Kit (KitName), with KitName as Primary Key

Locus (LocusID, KitName), with Primary Keys as LocusID, KitName and Foreign Key referencing KitName in Kit Table

Allele (AlleleID, LocusID), with AlleleID, LocusID as Primary Keys

Population (PopulationName), with PopulationName as Primary Key

Frequency (PopulationName, LocusID, AlleleID, Frequency), with PopulationName, LocusID, AlleleID as Primary Keys and Foreign referencing the PopulationName in Population Table and LocusID, AlleleID in Allele Table.

5.3 Database Integration with the Software

SQLite database has been connected by using the Java JDBC library. The database had to be integrated at various points in the application. After identifying the sections and their specific queries needed for the events triggered by a user, the code was divided into basic CRUD Operations for **C**reate, **R**etrieve, **U**pdate, and **D**elete.

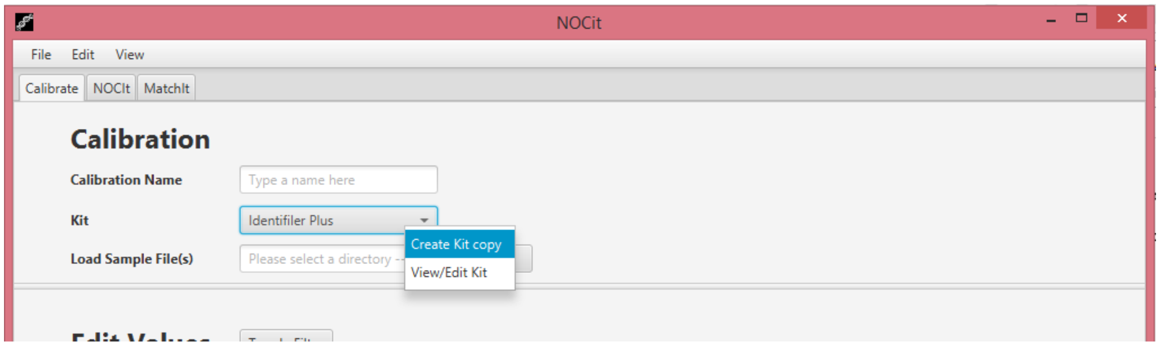
Example of code snippet for establishing Database Connection:

```
public static Connection connect() {

           try {
             Class.forName(DRIVER);
             SQLiteConfig config = new SQLiteConfig();
             config.enforceForeignKeys(true);
             Connection c = DriverManager.getConnection(DB_URL, config.toProperties());
             return c;
              } catch ( Exception e ) {
                      ~~Dialogs~~.create()
            .title(Constants.DATABASE_ERROR)
            .message("Error connecting to the Frequency Database.")
            .showError();
                      System.exit(0);
                      return null;


              }
      }
```

In the above code, Class.forName() function loads the JDBC driver class which contains the code for driver registration in a static block, using the Reflection API (java.lang.reflect). The next two lines switched on the Foreign Keys in SQLite as it needs to be manually done on every connection with a SQLite database. A connection is then returned by the getConnection method that gets the Database URL as its parameter. This connection is then returned to the code calling this function. This gets called from the UIController class that initiates the process when the application starts.

5.3.1 Calibration Interface with Database

For the Calibration, the first step is to select the Kit. The Kits present in the database are populated in the dropdown. There is a context menu attached with the Kit selection that is shown in Figure 5.2.



Calibration Name: Type a name here
Kit: Identifiler Plus
Load Sample File(s): Please select a directory

Figure 5.3: Kit edit context menu.

The Create Kit Copy creates a Duplicate of the Kit with all its Loci, Allele and Frequency values and inserts into the database. The dropdown for the kit auto-populates and shows the newly kit added as shown in Figure 5.3.

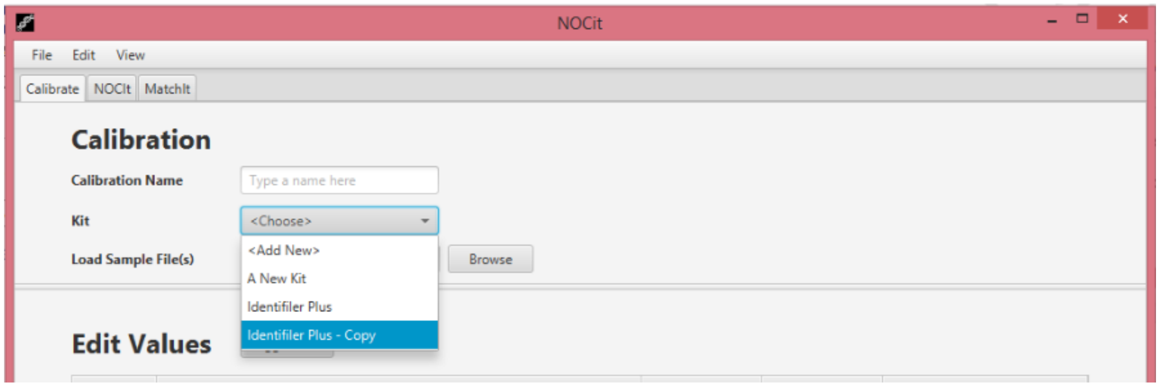


Figure 5.3: Dropdown menu for Kit selection.

The second option pops up a simple interface to edit the Kit name and the Loci present it the Kit. A context menu for each of the Loci present provides another interface to edit the Alleles present in the specific Locus. There are validation checks for all the values for Loci, Alleles, and Kit names before being saved to the database, including whether the kit has locus and alleles or not.

Figure 5.4 shows the Kit editor interface, which allows the user to view all the Loci present for the Kit with a context menu for viewing the Alleles present at each Locus.

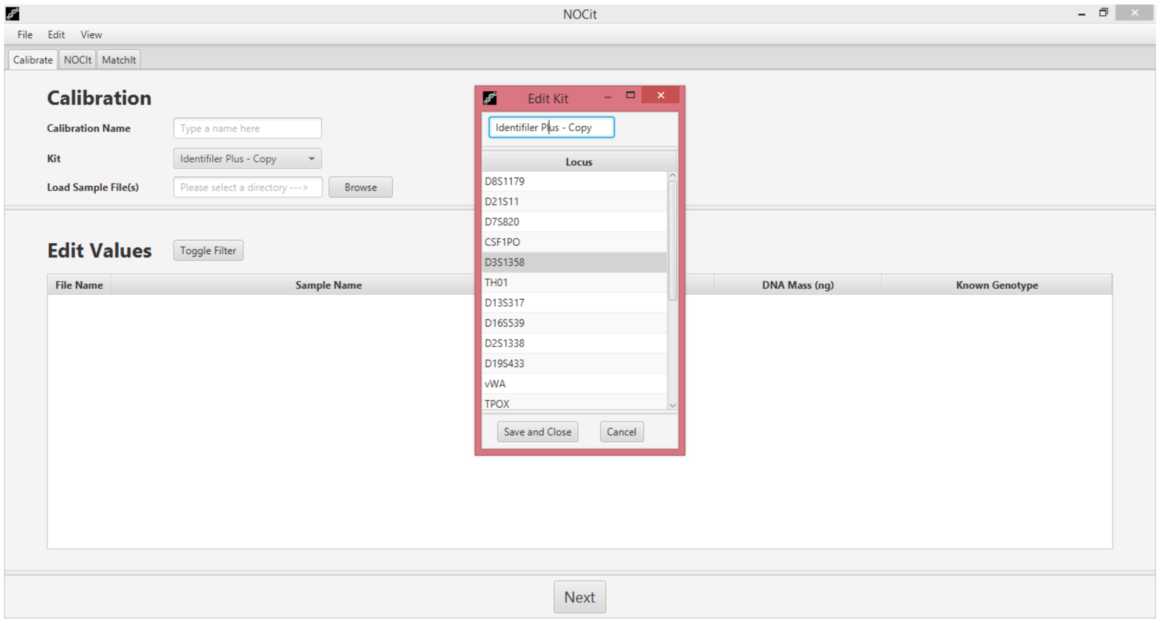


Figure 5.4: Pop-up interface for viewing the Kit and the Loci and editing the Loci and their Alleles.

Figure 5.5 shows the interface for editing the Alleles inside each Locus on top of the edit Kit interface.

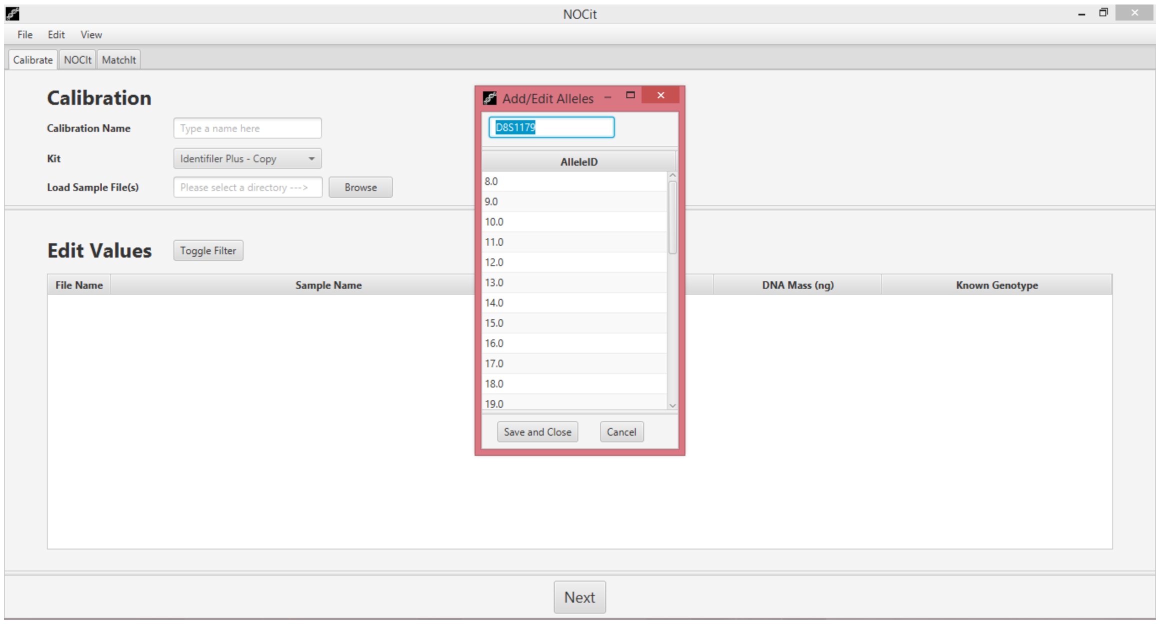


Figure 5.5: Pop-up interface for viewing and editing the Alleles present at each Locus for the Kit.

Figure 5.6 shows the interface for adding a new Kit, which is the first option in the dropdown.

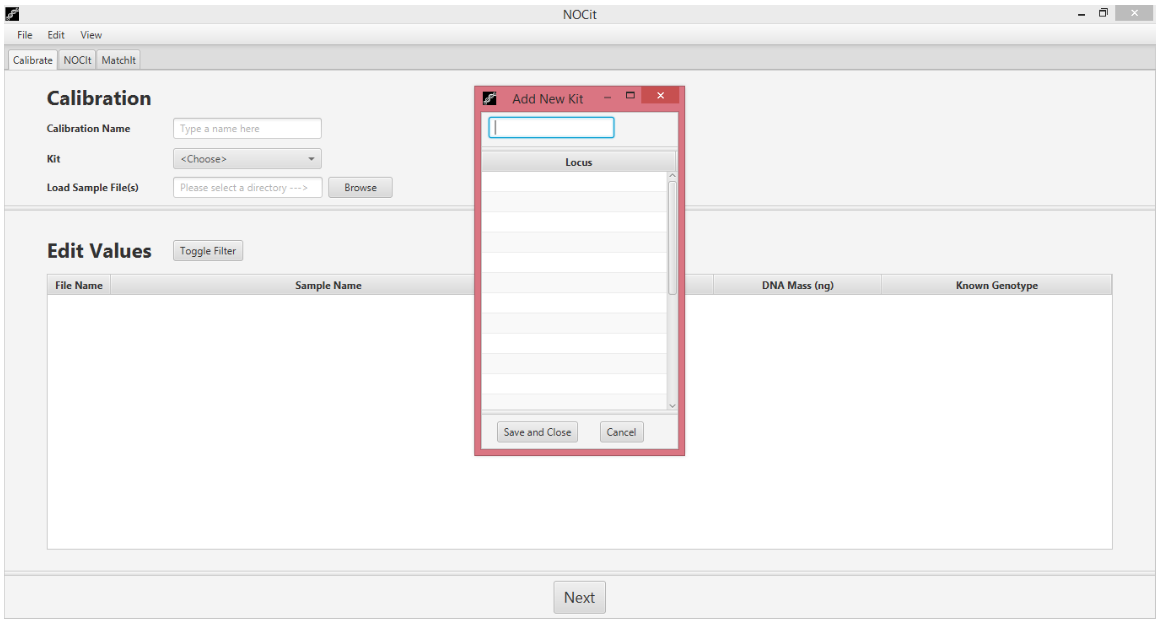


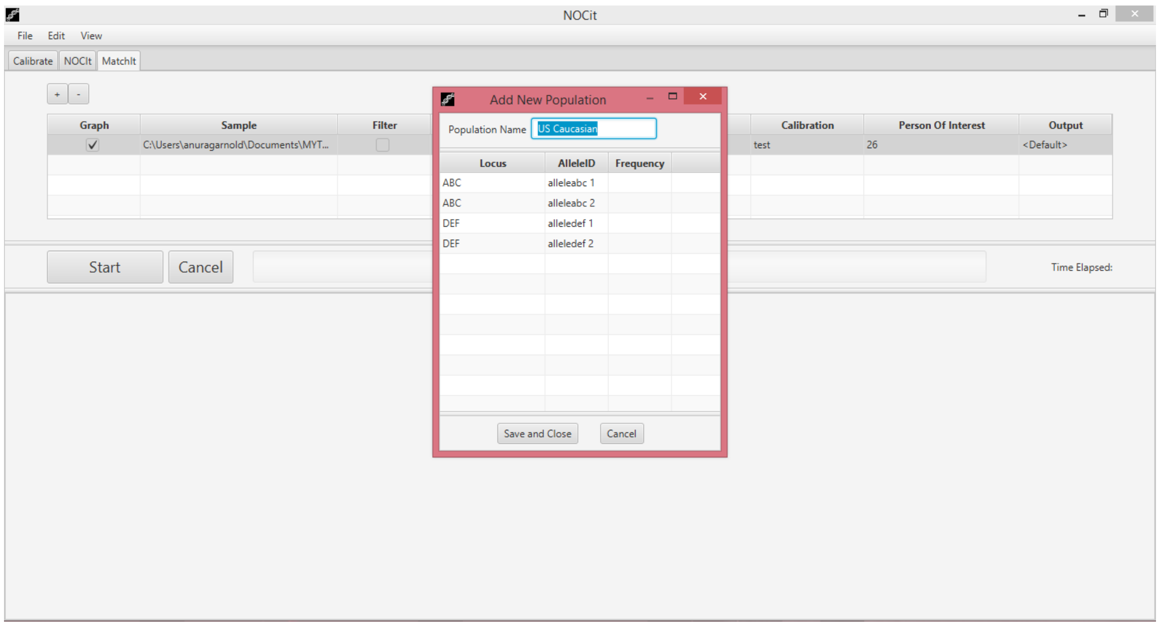
Figure 5.6: Pop-up interface for adding a new Kit.

Once the Kit has been selected, the user is supposed to select the Folder where all the sample files for calibration exist. All the sample files are validated against the Loci present in the database for the selected kit. Once the samples have been imported, the kit selection becomes un-editable. The same validation checks are in place for the bulk import of Genotypes (see Section 5.3.3).

5.3.2 MatchIt with Database

The various use cases that originate depending on the Kit selected or edited and if new one is added, the frequencies for the alleles in the kit are addressed in the MatchIt and NOCit interface when they are started. The algorithm checks for the presence of frequencies for all the alleles in the loci present in the kit selected, and if they are missing, then a similar interface pops up for the user to enter the missing frequencies. The validation checks for the frequencies whether they add up to 1.0 and for their acceptable value in float. A consistent looking pop up is also used if the user clicks on the Add New Population in the MatchIt or NOCit.

Figure 5.7 shows the interface for entering the missing frequencies for the selected Population type.



Graph	Sample	Filter	Calibration	Person Of Interest	Output
✓	C:\Users\anuragarnold\Documents\MYT...		test	26	<Default>

Population Name: US Caucasian

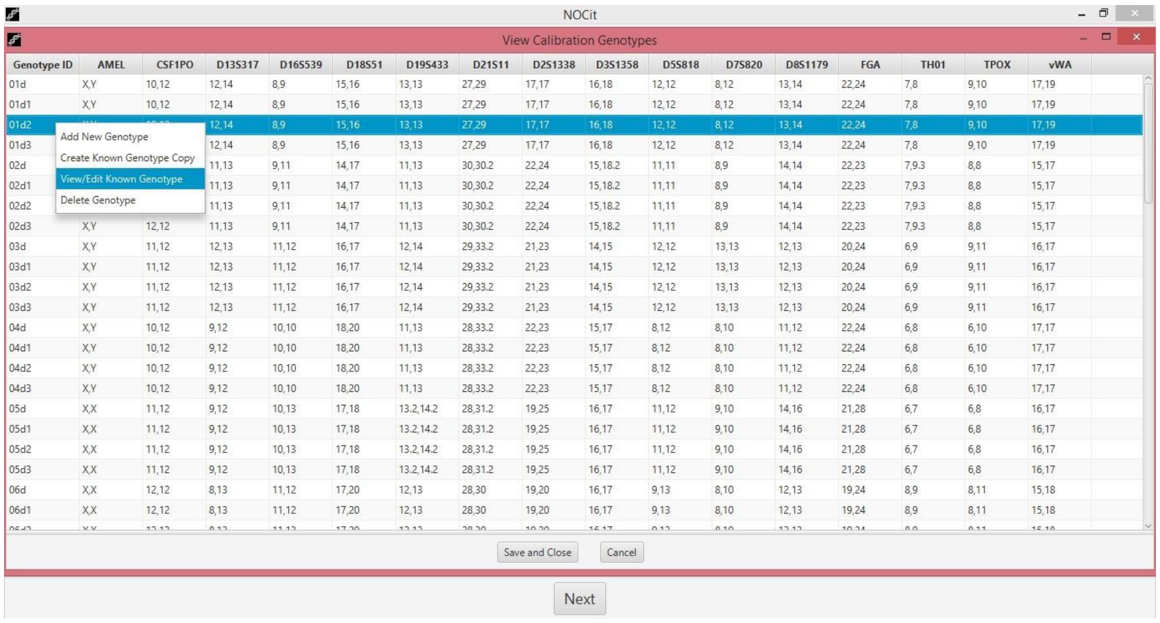
Locus	AlleleID	Frequency
ABC	alleleabc 1	
ABC	alleleabc 2	
DEF	alleledef 1	
DEF	alleledef 2	

Figure 5.7: Pop-up interface for entering the allele frequencies for the selected population in MatchIt.

5.3.3 General Menu Options with Database

Another problem that needed to be addressed was, even though there is a very simple way to enter the genotype sample data for each locus, it would take a long time for a user to input a lot of genotypes at once. So I designed an algorithm for a bulk import of all the Calibration Genotypes and Person of Interest Genotypes from an external csv file, through the Menu Option under File. These two imports have their own viewers in the View Menu Option.

Figure 5.8 shows the simple and elegant viewer for the Calibration Genotypes that have been imported by a user. A similar viewer is for the Person of Interest Genotypes as well.



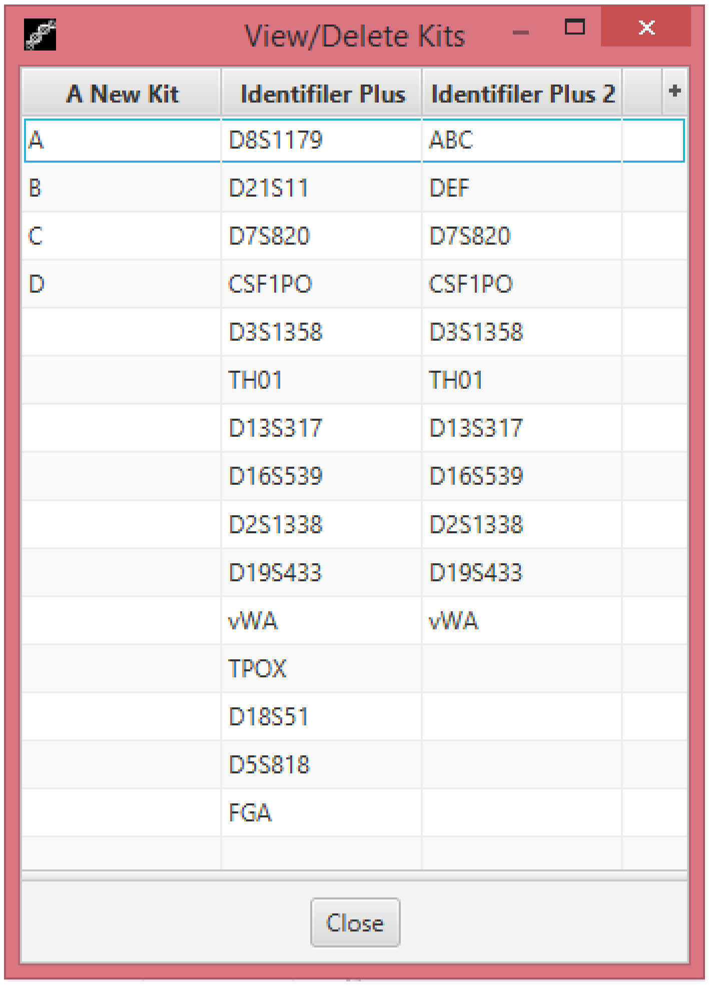
Genotype ID	AMEL	CSF1PO	D13S317	D16S539	D18S51	D19S433	D21S11	D2S1338	D3S1358	D5S818	D7S820	D8S1179	FGA	TH01	TPOX	vWA
01d	X,Y	10,12	12,14	8,9	15,16	13,13	27,29	17,17	16,18	12,12	8,12	13,14	22,24	7,8	9,10	17,19
01d1	X,Y	10,12	12,14	8,9	15,16	13,13	27,29	17,17	16,18	12,12	8,12	13,14	22,24	7,8	9,10	17,19
01d2			12,14	8,9	15,16	13,13	27,29	17,17	16,18	12,12	8,12	13,14	22,24	7,8	9,10	17,19
01d3			12,14	8,9	15,16	13,13	27,29	17,17	16,18	12,12	8,12	13,14	22,24	7,8	9,10	17,19
02d			11,13	9,11	14,17	11,13	30,30.2	22,24	15,18.2	11,11	8,9	14,14	22,23	7,9.3	8,8	15,17
02d1			11,13	9,11	14,17	11,13	30,30.2	22,24	15,18.2	11,11	8,9	14,14	22,23	7,9.3	8,8	15,17
02d2			11,13	9,11	14,17	11,13	30,30.2	22,24	15,18.2	11,11	8,9	14,14	22,23	7,9.3	8,8	15,17
02d3	X,Y	12,12	11,13	9,11	14,17	11,13	30,30.2	22,24	15,18.2	11,11	8,9	14,14	22,23	7,9.3	8,8	15,17
03d	X,Y	11,12	12,13	11,12	16,17	12,14	29,33.2	21,23	14,15	12,12	13,13	12,13	20,24	6,9	9,11	16,17
03d1	X,Y	11,12	12,13	11,12	16,17	12,14	29,33.2	21,23	14,15	12,12	13,13	12,13	20,24	6,9	9,11	16,17
03d2	X,Y	11,12	12,13	11,12	16,17	12,14	29,33.2	21,23	14,15	12,12	13,13	12,13	20,24	6,9	9,11	16,17
03d3	X,Y	11,12	12,13	11,12	16,17	12,14	29,33.2	21,23	14,15	12,12	13,13	12,13	20,24	6,9	9,11	16,17
04d	X,Y	10,12	9,12	10,10	18,20	11,13	28,33.2	22,23	15,17	8,12	8,10	11,12	22,24	6,8	6,10	17,17
04d1	X,Y	10,12	9,12	10,10	18,20	11,13	28,33.2	22,23	15,17	8,12	8,10	11,12	22,24	6,8	6,10	17,17
04d2	X,Y	10,12	9,12	10,10	18,20	11,13	28,33.2	22,23	15,17	8,12	8,10	11,12	22,24	6,8	6,10	17,17
04d3	X,Y	10,12	9,12	10,10	18,20	11,13	28,33.2	22,23	15,17	8,12	8,10	11,12	22,24	6,8	6,10	17,17
05d	X,X	11,12	9,12	10,13	17,18	13.2,14.2	28,31.2	19,25	16,17	11,12	9,10	14,16	21,28	6,7	6,8	16,17
05d1	X,X	11,12	9,12	10,13	17,18	13.2,14.2	28,31.2	19,25	16,17	11,12	9,10	14,16	21,28	6,7	6,8	16,17
05d2	X,X	11,12	9,12	10,13	17,18	13.2,14.2	28,31.2	19,25	16,17	11,12	9,10	14,16	21,28	6,7	6,8	16,17
05d3	X,X	11,12	9,12	10,13	17,18	13.2,14.2	28,31.2	19,25	16,17	11,12	9,10	14,16	21,28	6,7	6,8	16,17
06d	X,X	12,12	8,13	11,12	17,20	12,13	28,30	19,20	16,17	9,13	8,10	12,13	19,24	8,9	8,11	15,18
06d1	X,X	12,12	8,13	11,12	17,20	12,13	28,30	19,20	16,17	9,13	8,10	12,13	19,24	8,9	8,11	15,18

Figure 5.8: Interface for viewing all the Calibration genotypes or the POI genotypes.

All the allele values are editable in the Genotype Viewer. The values of 2 alleles are validated against the values present in the database.

The second problem that was addressed was to have the option to view and edit the Kits present in the database all at once. The whole kit can be deleted from the database through the context menu option present in the interface. The dropdown for kit selection auto populates after the deletion. Of course the option will not be able to delete if the kit is being used.

The View Kits option in the View Menu Option can be seen in the Figure 5.9.

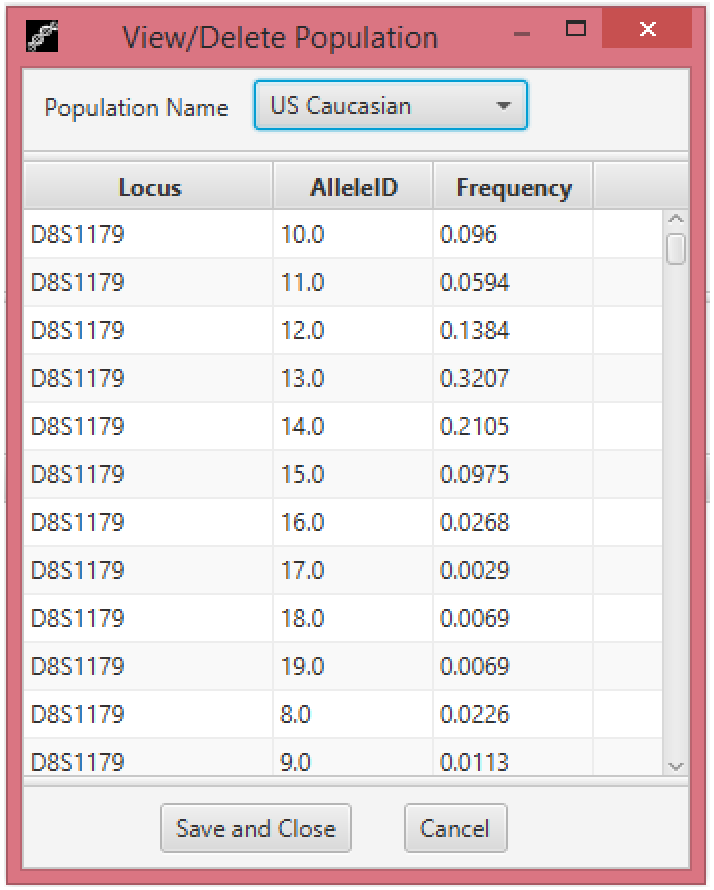


A New Kit	Identifiler Plus	Identifiler Plus 2
A	D8S1179	ABC
B	D21S11	DEF
C	D7S820	D7S820
D	CSF1PO	CSF1PO
	D3S1358	D3S1358
	TH01	TH01
	D13S317	D13S317
	D16S539	D16S539
	D2S1338	D2S1338
	D19S433	D19S433
	vWA	vWA
	TPOX	
	D18S51	
	D5S818	
	FGA	

Figure 5.9: Pop-up interface for viewing all the kits with their loci.

The screenshot shows the columns as the individual Kits present in the database, with their loci as their rows. The user has the option to hide the kit columns or see all of them as wanted.

A similar problem for Population view was also analyzed and implemented with the help of a Population Frequencies viewer through the View Menu Option. Figure 5.10 shows the screenshot of the Population Viewer.



Population Name: US Caucasian

Locus	AlleleID	Frequency
D8S1179	10.0	0.096
D8S1179	11.0	0.0594
D8S1179	12.0	0.1384
D8S1179	13.0	0.3207
D8S1179	14.0	0.2105
D8S1179	15.0	0.0975
D8S1179	16.0	0.0268
D8S1179	17.0	0.0029
D8S1179	18.0	0.0069
D8S1179	19.0	0.0069
D8S1179	8.0	0.0226
D8S1179	9.0	0.0113

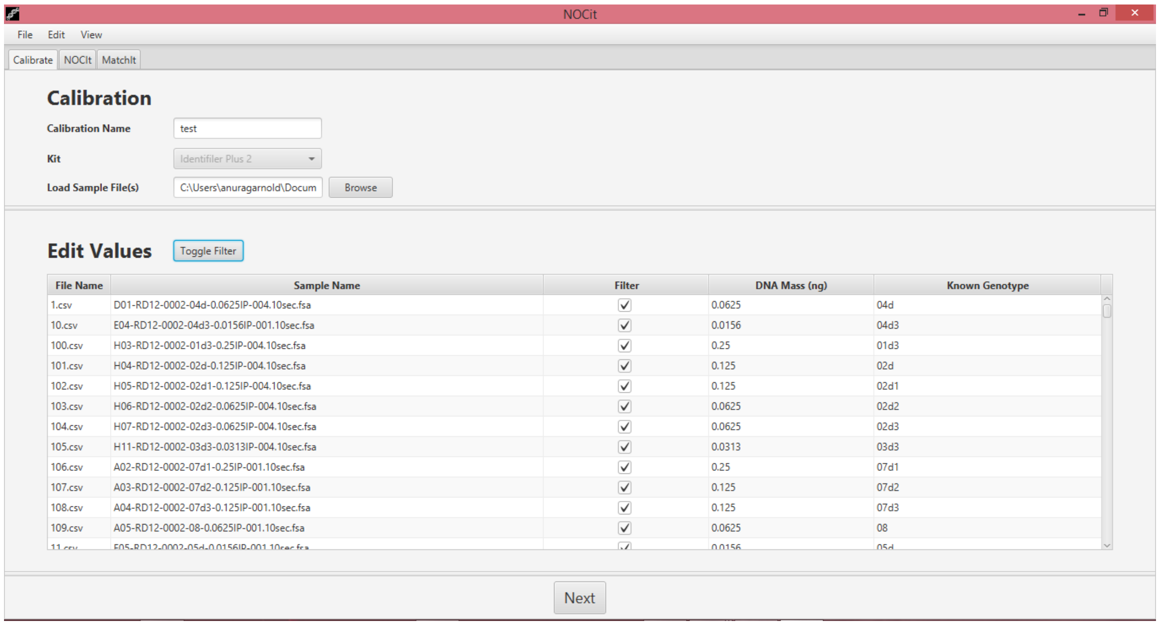
Figure 5.10: Interface for viewing the frequency data for each population with the context menu option for deleting any population as well.

The frequencies are editable in the interface for population viewer, for each of the population type present in the dropdown.

5.4 Screenshots and Results

The overall process from start to finish consists of the Calibration as the first step.

Figure 5.11 shows the first calibration tab where in the user selects the kit and inputs the sample data files and genotype files with all the parameters that are required to generate calibration file and its graphs.



Calibration

Calibration Name: test

Kit: Identifiler Plus 2

Load Sample File(s): C:\Users\anuragarnold\Docum

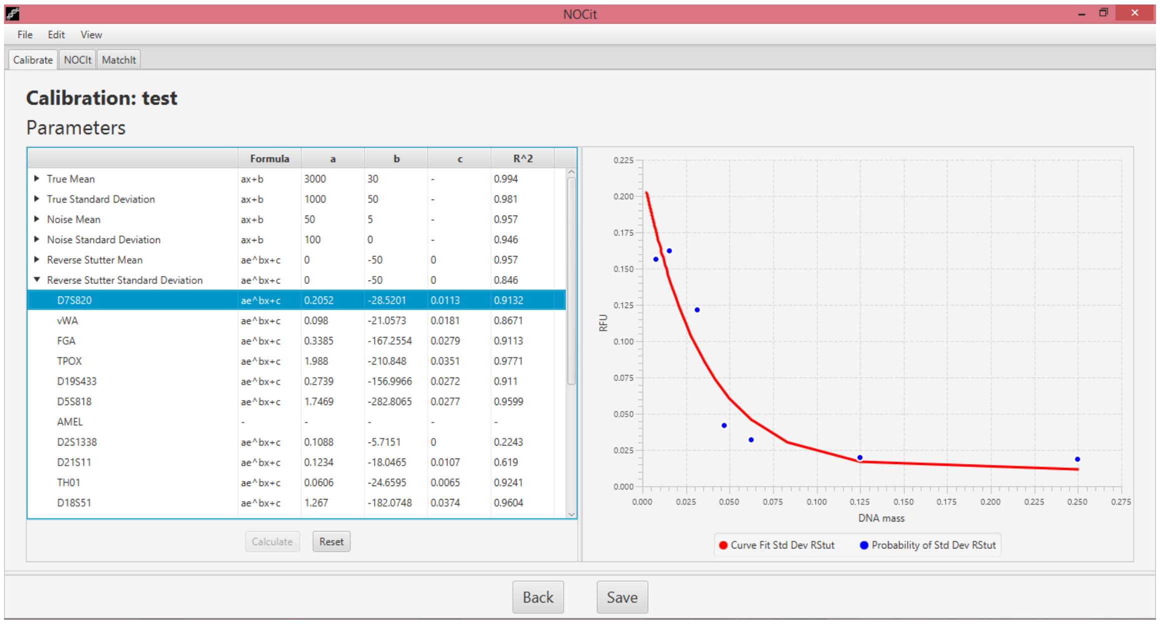
Edit Values

File Name	Sample Name	Filter	DNA Mass (ng)	Known Genotype
1.csv	D01-RD12-0002-04d-0.0625IP-004.10sec.fsa	✓	0.0625	04d
10.csv	E04-RD12-0002-04d3-0.0156IP-001.10sec.fsa	✓	0.0156	04d3
100.csv	H03-RD12-0002-01d3-0.25IP-004.10sec.fsa	✓	0.25	01d3
101.csv	H04-RD12-0002-02d-0.125IP-004.10sec.fsa	✓	0.125	02d
102.csv	H05-RD12-0002-02d1-0.125IP-004.10sec.fsa	✓	0.125	02d1
103.csv	H06-RD12-0002-02d2-0.0625IP-004.10sec.fsa	✓	0.0625	02d2
104.csv	H07-RD12-0002-02d3-0.0625IP-004.10sec.fsa	✓	0.0625	02d3
105.csv	H11-RD12-0002-03d3-0.0313IP-004.10sec.fsa	✓	0.0313	03d3
106.csv	A02-RD12-0002-07d1-0.25IP-001.10sec.fsa	✓	0.25	07d1
107.csv	A03-RD12-0002-07d2-0.125IP-001.10sec.fsa	✓	0.125	07d2
108.csv	A04-RD12-0002-07d3-0.125IP-001.10sec.fsa	✓	0.125	07d3
109.csv	A05-RD12-0002-08-0.0625IP-001.10sec.fsa	✓	0.0625	08
11.csv	F05-RD12-0002-05d-0.0156IP-001.10sec.fsa	✓	0.0156	05d

Figure 5.11: First Calibration Tab

After the user clicks on the Next button, the user is supposed to click Calculate button to generate the calibration data in tabular form and, plots on the right side for each of the corresponding parameters from the tree table.

Figure 5.12 shows the second tab for the Calibration process.



Calibration: test

Parameters

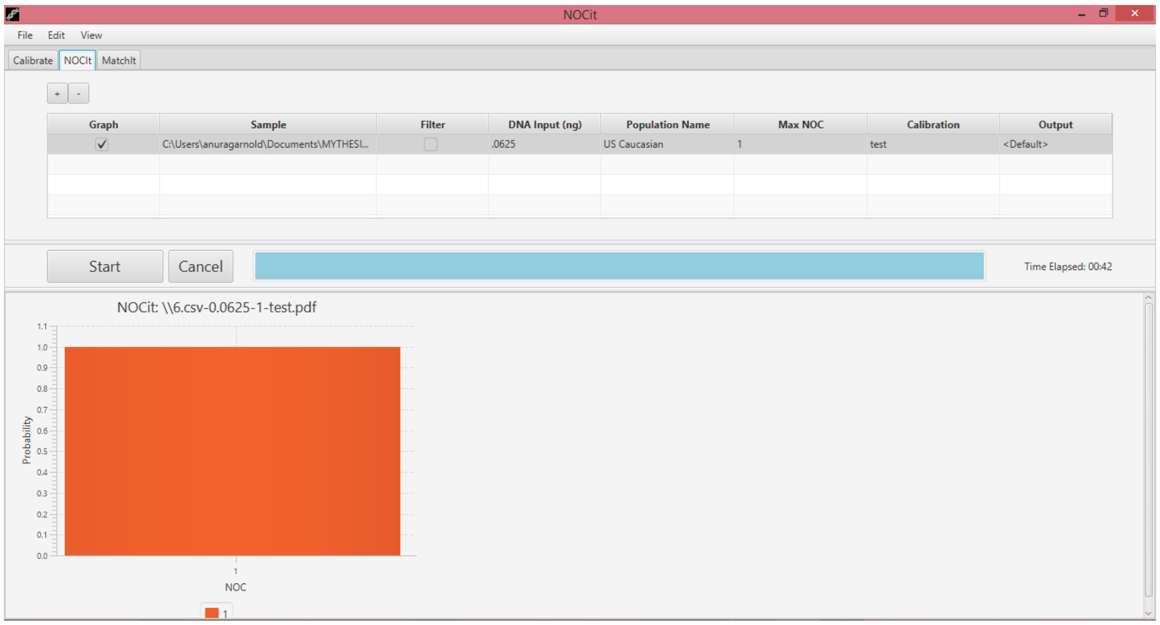
	Formula	a	b	c	R^2
▸ True Mean	ax+b	3000	30	-	0.994
▸ True Standard Deviation	ax+b	1000	50	-	0.981
▸ Noise Mean	ax+b	50	5	-	0.957
▸ Noise Standard Deviation	ax+b	100	0	-	0.946
▸ Reverse Stutter Mean	ae^bx+c	0	-50	0	0.957
▾ Reverse Stutter Standard Deviation	ae^bx+c	0	-50	0	0.846
D7S820	ae^bx+c	0.2052	-28.5201	0.0113	0.9132
vWA	ae^bx+c	0.098	-21.0573	0.0181	0.8671
FGA	ae^bx+c	0.3385	-167.2554	0.0279	0.9113
TPOX	ae^bx+c	1.988	-210.848	0.0351	0.9771
D19S433	ae^bx+c	0.2739	-156.9966	0.0272	0.911
D5S818	ae^bx+c	1.7469	-282.8065	0.0277	0.9599
AMEL	-	-	-	-	-
D2S1338	ae^bx+c	0.1088	-5.7151	0	0.2243
D21S11	ae^bx+c	0.1234	-18.0465	0.0107	0.619
TH01	ae^bx+c	0.0606	-24.6595	0.0065	0.9241
D18S51	ae^bx+c	1.267	-182.0748	0.0374	0.9604

Figure 5.12: Second Calibration tab with graphs and tree table

After the Calibration data has been generated, the user has the option of clicking on Save button to save the Calibration file for it to be later used in NOCit and MatchIt.

The NOCit and MatchIt Tabs are the next two tabs where the top section is used by the user to set up the input data, and the bottom pane is where the results with the graph are displayed once the calculations are done.

Figures 5.13 and 5.14 show the NOCit and MatchIt Tabs after the results have been generated.

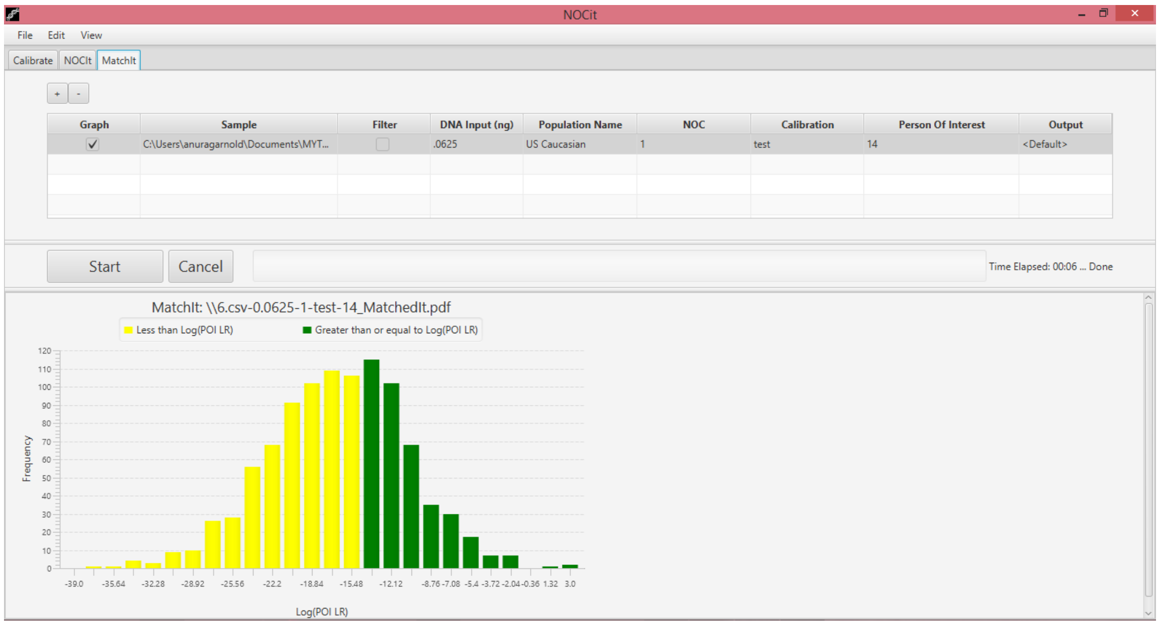


Graph	Sample	Filter	DNA Input (ng)	Population Name	Max NOC	Calibration	Output
✓	C:\Users\anuragarnold\Documents\MYTHESI...	☐	.0625	US Caucasian	1	test	<Default>

Time Elapsed: 00:42

NOCit: \\6.csv-0.0625-1-test.pdf

Figure 5.13: NOCit Tab and Interface



Graph	Sample	Filter	DNA Input (ng)	Population Name	NOC	Calibration	Person Of Interest	Output
✓	C:\Users\anuragarnold\Documents\MYT...	☐	.0625	US Caucasian	1	test	14	<Default>

Time Elapsed: 00:06 ... Done

MatchIt: \\6.csv-0.0625-1-test-14_MatchedIt.pdf

Figure 5.14: MatchIt Tab and Interface.

Conclusion

This work comprises of three parts, starting with the implementation of a filtering algorithm, user interface design for MatchIt and integration of a relational database with the software application.

The filtering algorithm is a small subset of the underlying algorithms, yet an integral part of the preprocessing stage for generating the calibration file before NOCit and MatchIt calculations. The user interface design for MatchIt has contributed in making the software use a smooth, consistent and user-friendly experience. The various interfaces connecting to the backend database make the application feature-rich with user options. This has also helped in attaining a concise way to structure and store the data used in the application in an organized manner.

With this work we have contributed substantially to the development of a number of components of the continuing research pipeline for forensic DNA analyses.

References

Butler, J.M. (2009). *Fundamentals of forensic DNA typing*, first ed., Associated Press.

Butler, J.M. (2014). *Advanced Topics in Forensic DNA Typing: Interpretation*, first ed., Academic Press.

DNA profiling, (n.d.). In Wikipedia. Retrieved October, 2015, from https://en.wikipedia.org/wiki/DNA_profiling

Murphy, Erin (2015). *Inside the Cell: The Dark Side of Forensic DNA*, Nation Books.

Riley, Donald E. (2005). *DNA Testing: An Introduction for Non-Scientists*, University of Washington.

STR analysis, (n.d.). In Wikipedia. Retrieved October, 2015, from https://en.wikipedia.org/wiki/STR_analysis

Swaminathan, H., Grgicak, C.M., Medard, M., Lun, D.S. (2015). *NOCIt: A computational method to infer the number of contributors to DNA samples analyzed by STR genotyping*, Forensic Sci Int Genet. 16 172-180. DOI: 10.1016/j.fsigen.2014.11.010.

SWGDAM Interpretation Guidelines for Autosomal STR Typing http://swgdam.org/Interpretation_Guidelines_January_2010.pdf