DEVELOPMENT OF A COMPUTATIONAL TOOL FOR FORENSIC DNA ANALYSIS

By ABHISHEK GARG

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

October 2015

THESIS ABSTRACT

Development of a computational tool for Forensic DNA Analysis

By ABHISHEK GARG

Thesis Director
Dr. Desmond S. Lun

Forensic DNA analysis uses repetitive sequences in the human genome called Short Tandem Repeats (STRs) for human identification. This study has contributed to the building, design, and development of a fast and easy to use software package to calculate the *a posteriori* probability (APP) on the number of contributors in an STR profile and to calculate the likelihood ratio (LR), a statistic that conveys the strength of a match for a given suspect, as well as the distribution of the LR over random non-contributors. This research specifically deals with the (1) design and implementation of an algorithm for MatchIt (the component that calculates the LR and its distribution); (2) optimization the code to reduce running time; and (3) development of a user-friendly interface.

ACKNOWLEDGEMENTS

I would never have been able to finish my dissertation without the guidance of my advisor, support of my committee members, help from friends and full support from my family.

I would like to express my deepest gratitude to my advisor, Dr. Desmond Lun, for his excellent guidance, expertise, patience, and providing me with an excellent atmosphere for doing research. I would like to thank Dr. Dawei Hong, Dr. Michael Palis, Dr. J C Birget and Dr. Sunil Shende who helped me to develop my background in optimization methods, parallel algorithms, numerical methods and big data algorithms.

I would like to thanks Harish Swaminathan, who was always a good friend, was always willing to help and give his best suggestions. Many thanks to Mark Moore and Anurag Arnold for helping in the implementation of user design. My research would not have been possible without their help.

Finally, I would like to say thanks to Dr. Suneeta Ramaswami for giving me an opportunity to be a part of Department of Computer Science at Rutgers Camden.

Table of Contents

Thesis Abstract II

Acknowledgements III

Table of Contents iv

List of Figures v

Introduction 1

Purpose 3

Design and Implementationof MatchIt Algorithm 4

3.1 Introduction 4

3.2 MatchIt Algorithm 5

3.2.1 LR numerator calculation 6

3.2.2 LR distribution and p-value calculation 9

3.2.3 LR denominator calculation 11

3.3 Testing and Results 11

Code Optimization using various techniques 16

4.1 Introduction 16

4.2 Example of one code snippet 17

4.3 Results 18

User interface design and development 20

5.1 Introduction 20

5.2 Design and Implementation 20

5.3 Results 20

Appendix A 24

REFERENCES 26

List of Figures

Figure 1 Sample electropherogram with peak heights at each allele........ 5
Figure 2 Calibration data of true peak mean at locus D7S820, distribution ax+b, R square value .9929 12
Figure 3 Calibration data of reverse stutter peak mean at locus D7S820, distribution a*ebx +c, R square value .9956........ 12
Figure 4 Calibration data of dropout at locus D7S820, distribution a*ebx , R square value .9946 mass........ 13
Figure 5 Graph showing results for MatchIt using 1p samples 14
Figure 6 Graph showing results for MatchIt for 2p sample 14
Figure 7 Graph showing results for MatchIt for 3p sample 15
Figure 8 Reduction in running time of base code snippet with each optimization .. 20
Figure 9 Software efficiency percent gain in different versions........ 19
Figure 10 First tab of software - Calibration........ 19
Figure 11 Second window of Calibration tab with the resuling data 21
Figure 12 NOCit tab with the result........ 22
Figure 13 MatchIt tab........ 23

Introduction

Starting from mid 90s, Short Tandem Repeats (STRs) have been used in the field of human identification for forensic purposes [1]. STRs are repetitive sequences that are 1–7 base pairs in length and scattered throughout the human genome. An STR DNA profile developed from a biological sample (like saliva, semen, blood, etc.) collected at a crime scene is either compared with that of a person of interest (POI) or run against a database to check for a match. The Scientific Working Group on DNA Analysis Methods (SWGDAM) recommends that forensic reports include a statement regarding the assumption made about the number, or the minimum number of contributors, to the sample being investigated [2]. The number of contributors to a crime scene sample is generally unknown and must be estimated by the analyst based on the electropherogram obtained.

An assumption about the number of contributors is needed when determining whether a known should be excluded as a contributor to an item of evidence. Changing the number of contributors could lead to different conclusions about whether to include or exclude an individual as a contributor to the sample. Further, an assumption about the number of contributors to a sample is needed to calculate a match statistic, called the Likelihood Ratio [3]. A statistic that is commonly used internationally and is gaining acceptance in the United States.

The Likelihood Ratio (LR) 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.

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

Purpose

The purpose of the current study was to develop a software package that estimates the number of contributors to an evidence sample and subsequently calculates a LR for person of interest based the given DNA sample and the number of contributors in an accurate, reliable and efficient manner. This work is specifically focused on the following three aspects of mixture interpretation:

1. Design and implementation of an algorithm for MatchIt
2. Code optimization to reduce running time
3. User interface design and development

Design and Implementation of MatchIt Algorithm

3.1 Introduction

The software is divided mainly in three parts as 1) Calibration, 2) NOCIt, and 3) MatchIt.

Characterization of the peak heights is done by using single source calibration profiles with known genotypes obtained from samples amplified from a wide range of input DNA masses. Absolute DNA (extracted from 28 individuals) quantification was performed using real-time PCR and the Quantifiler® Duo™ Quantification kit according to the manufacturer's recommended protocol and one external calibration curve [4, 5]. The extracted DNA was amplified using the manufacturer's recommended protocol for AmpFℓSTR® Identifiler® Plus Amplification Kit (Life Technologies, Inc.) [6]. A fragment analysis was performed using GeneMapper IDX v1.1.1 (Life Technologies, Inc.) and an RFU threshold of one. A threshold of 1 RFU was used in order to capture all peak height information (i.e., the allelic peaks, baseline noise and stutter peaks) in the signal. Known artifacts such as pull-up, spikes, -A, and artifacts due to dye dissociation were manually removed while generating the sample data. Refer to figure 1 for a sample electropherogram with peak heights at each allele. For a detailed description of how the calibration samples were created, refer to [7].

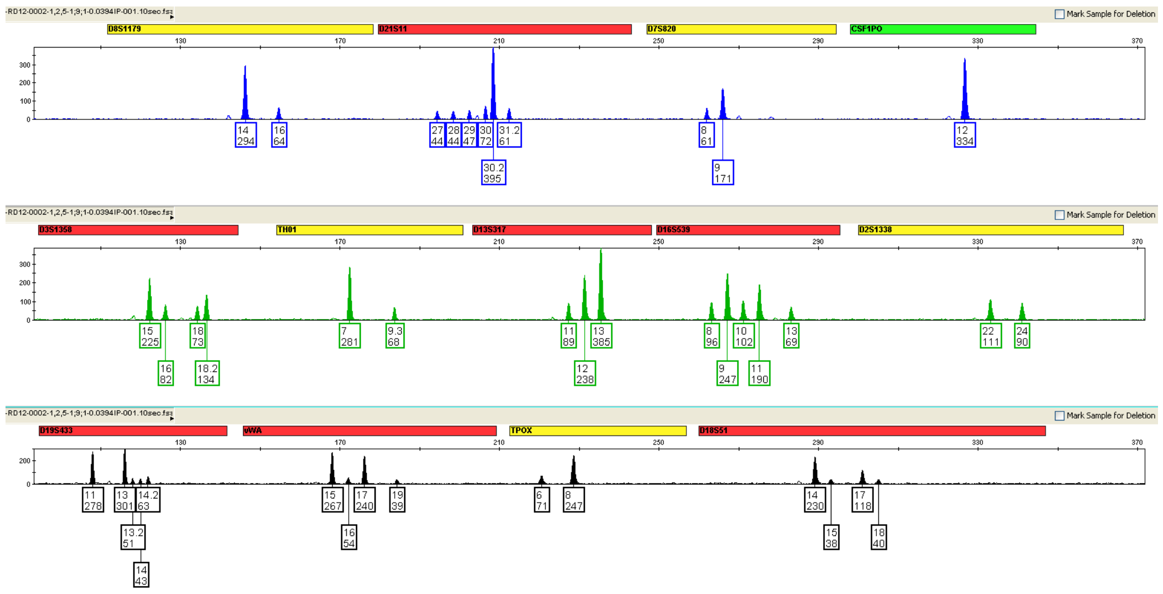


Figure 1 Sample electropherogram with peak heights at each allele

NOCIt is a computational tool that calculates the APP on the number of contributors to a DNA sample [7].

MatchIt uses single source samples with known genotypes and calculates a LR and a *p*-value for a specified POI on a question sample. It is a fully continuous method that works by modeling the peak heights observed in a calibration data set consisting of single source samples with known genotypes. It accounts for dropout and stutter (both reverse and forward), two common artifacts observed in low template samples [8]. Additionally, MatchIt also computes a *p*-value for the LR by sampling a large number of random genotypes from the population.

3.2 MatchIt Algorithm

The Likelihood Ratio (LR) is defined as:

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

In practice, n_p and n_d can be chosen by the prosecution and the defense to maximize their respective probabilities and there is no necessity for n_p to be equal to n_d. However, we have developed MatchIt to use the same number of contributors in both the numerator and denominator to calculate the LR. It should be noted that the method could be extended to work on different assumptions on the number of contributors. Moving forward, we omit the notation n for the sake of brevity. We note that for purposes of this work, $n_p = n_d$ in all cases presented herein, and we use the known, and thus the true n to test the capabilities of MatchIt.

For this study, we use the following hypotheses for H_p and H_d:

H_p:The evidence is a mixture of the genotype profile of a suspect (s) and the profiles of $n - 1$ other unknown, unrelated contributors, whom we term for the purpose of this paper as "the interference contributors".

H_d: The evidence is from n unknown individuals unrelated to the suspect.

In most cases, the value of the LR is very large (or very small) and it is easier to work with log(LR). Hence we have:

$$\log\big(LR(s)\big) = \log(Pr(E|R = s, U^{n-1})) - \log(\Pr(E|U^n)),$$

where $U^i = \{U_1, \dots, U_i\}$ are the random genotypes of i contributors and R is the random genotype of a single contributor, whether it be a true contributor, or non-contributor.

3.2.1 LR numerator calculation

Our algorithm assumes a constant mixture ratio at all the loci. The mixture ratio specifies the proportion of the total template mass contributed by each contributor to the sample. The underlying mixture ratio of an evidence sample is unknown and needs to be described by a model in order to compute a continuous LR. A constant mixture ratio model assumes that the mixture ratio is the same at all the markers, whereas a variable mixture model accounts for the possibility of the mixture ratio being different at the various markers. Both models are reasonable and are used in existing continuous methods to compute the LR. Perlin et al [9] assign a uniform prior probability for the template mixture weight and construct its probability distribution by drawing individual locus weights using a multivariate normal distribution. Cowell et al [10] and Puch-Solis et al [11] use a constant mixture ratio model and implement a discrete approximation over the interval (0,1) by assigning a uniform prior. Taylor et al [12] use the variable model and assume the mixture weights to be independent across the loci. Since we adopt the constant mixture ratio approach, we integrate over all possible mixture ratios to calculate the probability of observing the evidence:

$$\Pr(E|R = s, U^{n-1}) = \int_{\theta\in\Delta^{n-1}} Pr(E|\Theta = \theta, R = s, U^{n-1})\, f_\Theta(\theta),$$

where Θ is the vector with components Θ_i, the mixture proportion of each contributor $i \in \{1, \dots, n_{\max}\}$; $\Delta^{n-1} = \{(\Theta_1, \dots, \Theta_n) \in R^n | \sum_{i=1}^{n} \Theta_i = 1, \Theta_i > 0\forall i\}$ is the unit $n-1$ simplex; and f_Θ is the probability density function of Θ, which we assume to be uniform over Δ^{n-1}. For n = 1, Δ^{n-1} consists of the single element $\{1\}$. For mixtures, we implement the integration over Δ^{n-1} by dividing it into equal-sized subsets and representing each subset with its centroid, resulting in a discrete sum.

To do this, we performed k-means clustering in Python (Python Software Foundation, Beaverton, Oregon). k-means clustering is an algorithm used to partition observations into a set of clusters by repeated minimization of the distance from an observation to the centroid of its cluster [13]. For n = 2, the space was divided into 9 equally sized clusters, while for n = 3, 12 clusters were used.

Let L be the set of all loci in the evidence sample, E_l be the evidence at locus l,U_l^{n-1} be the genotype of the interference contributors at locus l and s_l be the genotype of the suspect at locus l. The STR loci used for forensic DNA analysis are assumed to be in linkage equilibrium and independent of each other [14]. Hence we obtain:

$$\Pr(E|\Theta = \theta, R = s, U^{n-1}) = \prod_{l \in L} Pr(E_l|\Theta = \theta, R_l = s_l, U_l^{n-1}).$$

The prosecution's hypothesis states that the profile is made of the suspect's contribution plus the contribution from $n - 1$ other random, unrelated contributors. Since there are many possibilities for the genotype of these interference contributors at each locus and going over each case would take a large amount of time, we calculate $Pr(E_l|\Theta = \theta, R_l = s_l, U_l^{n-1})$ using importance sampling.

Importance sampling is a Monte Carlo sampling algorithm in which, instead of sampling directly from the target distribution, samples are generated from a different distribution that is easier to sample from [15]. To take into account the fact that the samples have come from the 'wrong' distribution, weights are introduced to adjust the 'importance' of each sample. For the problem at hand, instead of sampling using the allele frequency distribution, we generate samples of the interference genotypes using the peak height distribution observed at the locus. The reason for sampling from the peak height distribution is that this method is faster and requires fewer samples for convergence than the method that samples from the allele frequency distribution.

Let J be the number of interference samples used. Now we obtain:

$$Pr(E_l|\Theta = \theta, R_l = s_l, U_l^{n-1}) = \frac{\sum_{i=1}^{J} Pr\left(E_l|U_{l_i}^{n-1} = u_{l_i}^{n-1}, \Theta = \theta, R_l = s_l\right) w_i}{\sum_{i=1}^{J} w_i},$$

where $w_i = P\left(U_{l_i}^{n-1}\right)/Q\left(U_{l_i}^{n-1}\right)$ is the weight of sample i; $P\left(U_{l_i}^{n-1}\right)$ is the probability of the interference genotypes under the allele frequency distribution; and $Q\left(U_{l_i}^{n-1}\right)$ is the probability of the interference genotypes under the peak height distribution. Since $u_{l_i}^{n-1}$ and s_l establish the true peaks in the signal (and by extension the stutter and noise peaks), $Pr\left(E_l \vee U_{l_i}^{n-1} = u_{l_i}^{n-1}, \Theta = \theta, R_l = s_l\right)$ is calculated using the parameters from the calibration data.

3.2.2 LR distribution and p-value calculation

Since the denominator of the LR is the same for all the random genotypes R, it is sufficient if we compare the numerator of the LR for R and s.

$$p\text{-}value(s) = Pr\left(Pr(E|R, U^{n-1}) \geq Pr(E|R = s, U^{n-1})\right).$$

During testing of MatchIt, we observed that because of floating-point precision, $Pr(E \vee R, U^{n-1})$ evaluated to 0 for many of the random genotypes R that fit the data poorly. As a result, we were able to eliminate those genotypes from the p-value calculation as a preliminary step. Formally, let R be the set of all genotypes. We define $R_1 = \{r \in R \vee Pr(E_l|R_l = r_l) \not\simeq 0 \text{ for all loci } l\}$ and $R_2 = \{r \in R \vee \exists \text{ locus } l \text{ s.t.} Pr(E_l|R_l = r_l) \simeq 0\}$, where $\simeq 0$ means "evaluates to 0 using double-precision 64-bit floating-point arithmetic". Thus, we have $R = R_1 \cup R_2 \wedge R_1 \cap R_2 = \emptyset$. We see that for all $r \in R_2, Pr(E|R = r) \simeq 0$. We omit the notation on U^{n-1} for the sake of brevity. We have:

$$p\text{-}value(s) = Pr(R \in R_1) \sum_{r \in R_1} Pr(\Pr(E|R = r) \mid \geq \Pr(E|R = s))Pr(R = r|R \in R_1)$$

$$+ Pr(R \in R_2) \textstyle\sum_{r \in R_2} Pr(E|R = r) \geq (Pr(E|R = s)Pr(R = r|R \in R_2)$$

We see that the second term is 0, provided $Pr(E \vee R = s)$ is greater than 0. Hence we get:

$$p\text{-}value(s) = Pr(R \in R_1) \sum_{r \in R_1} 1\big(Pr(E|R = r) \geq Pr(E|R = s)\big)Pr(R = r|R \in R_1),$$

where

$$1\big(Pr(E|R = r) \geq Pr(E|R = s)\big) = \begin{cases} 1, \text{if } Pr(E|R = r) \geq Pr(E|R = s), \\ \quad 0, \text{otherwise.} \end{cases}$$

We have:

$$Pr(R \in R_1) = \prod_{l \in L} \sum_{r_l \in \{r|Pr(E_l|R_l = r) \neq 0\}} Pr(R_l = r_l).$$

We compute the $p\text{-}value$ using Monte Carlo simulation. We generate M random genotypes $r^1, \ldots, r^M$ according to the distribution $Pr(R \vee R \in R_1)$ and calculate the p-value as:

$$p\text{-value}(s) = Pr(R \in R_1) \frac{\sum_{i=1}^{M} 1\left(Pr\big(E|R = r^i\big) \geq Pr(E|R = s)\right)}{M}$$

Increasing the value of M increases the accuracy of the p-value computed, but this also increases the run time and a hence a tradeoff has to be achieved between the two. In this study, we have used 1 billion or 109 random genotypes to compute the p-value.

In order to facilitate the computation of the p-value, as an initial step $Pr(E_l|\Theta = \theta, R_l = g_l, U_l^{n-1})$ is computed for all possible genotypes g_l at all loci l for all values of θ. Once this is done, for the p-value computation, 109 genotypes r^i are generated based on the allele frequencies. Since we know $Pr\left(E_l \middle| R_l = r_l^i\right)$ for all loci l, we can compute $Pr\left(E|R = r^i\right)$ as:

$$Pr\left(E \middle| R = r^i\right) = \int_{\theta \in \Delta^{n-1}} \prod_l Pr\left(E_l \middle| \Theta = \theta, R_l = r_l^i, U_l^{n-1}\right) f_\Theta(\theta).$$

3.2.3 LR denominator calculation

Let $\acute{R}$ be the genotype of an unknown contributor in the defense's hypothesis. The denominator of the LR can be written as:

$$Pr(E|U^n) = \sum_{\acute{r}} Pr(E|\bar{R} = \bar{r}, U^{n-1}) Pr(\bar{R} = \bar{r})$$

Since the number of possible values that $\bar{R}$ can take is large and summing over all of them is computationally intensive, we utilize the random genotypes r^i that are sampled for the p-value computation to compute the denominator of the LR as follows:

$$Pr(E|U^n) = Pr(R_1) \frac{\sum_{i=1}^{M} Pr\left(E|R = r^i\right)}{M}.$$

3.3 Testing and Results

The method was tested on 597 1-, 2- and 3-person profiles with total DNA template mass ranging from 0.016 to 0.25 Nano grams and an injection time of 10 sec. Calculated calibration data is then displayed in the software in graphical form. Few sample graphs are shown in the following figure 2, figure 3 and figure 4.

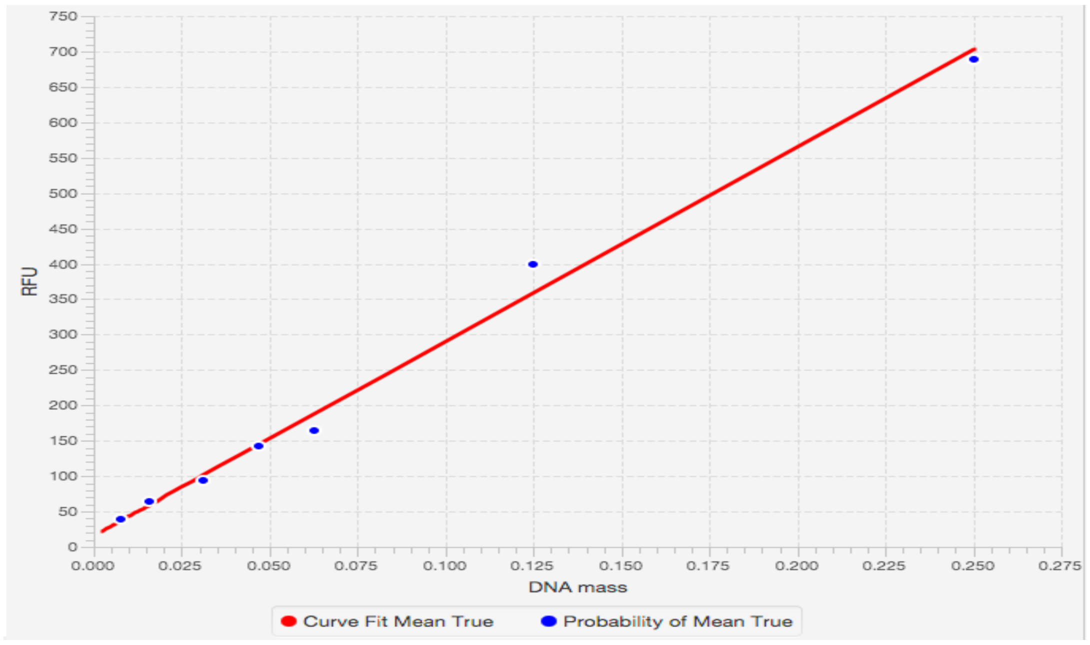


Figure 2 Calibration data of true peak mean at locus D7S820, distribution ax+b, R square value .9929

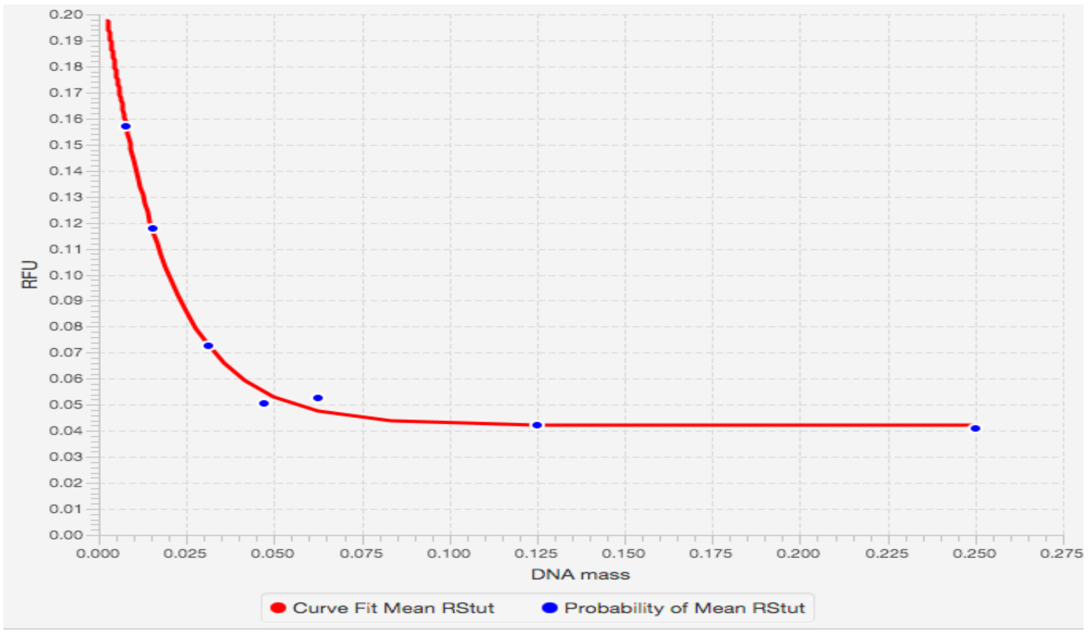


Figure 3 Calibration data of reverse stutter peak mean at locus D7S820, distribution a*e^{bx} +c, R square value .9956

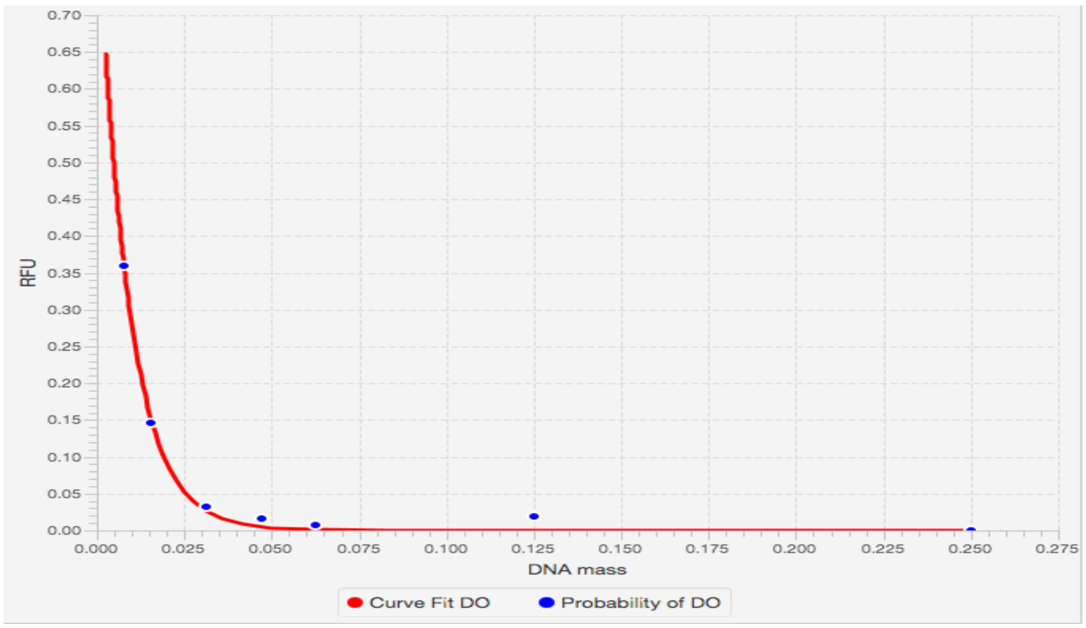


Figure 4 Calibration data of dropout at locus D7S820, distribution a*e^{bx} , R square value .9946 mass

We found that the amount of template DNA from the contributor had an impact on the LR – small LRs were associated with contributors having low template amounts and high levels of dropout and stutter. Since we used 10^9 samples to calculate the *p*-value, the lowest possible *p*-value that can be achieved is 10^9, and this was obtained in all the cases where the LR was greater than 10^8.

All the graphs shown below in Figure 5, 6 and 7 were tested for various sample files with varying DNA mass. Each sample was tested for actual contributors and 3 other non-contributors. And results were more or less as expected, which is for actual contributor p-value is -9 (as we are considering one billion samples) and for non-contributors it is close to zero.

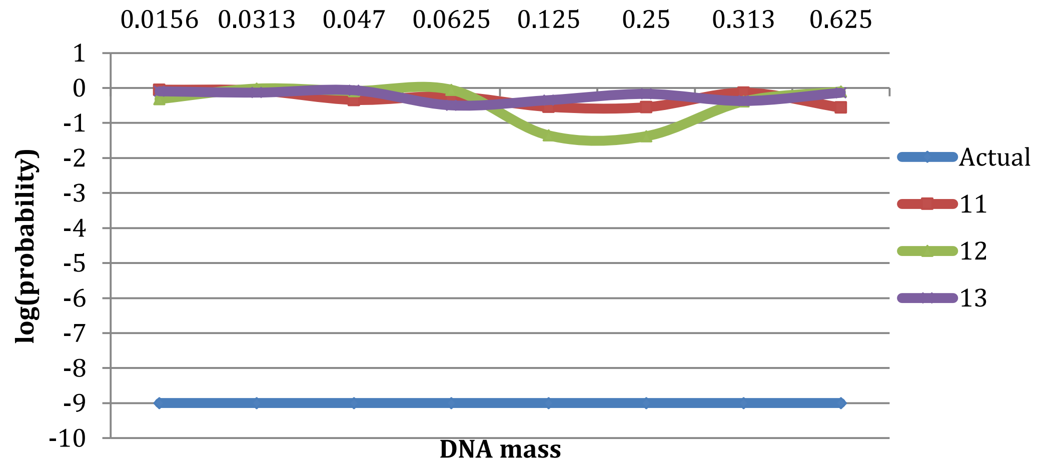


Figure 5 Graph showing results for MatchIt using 1p samples

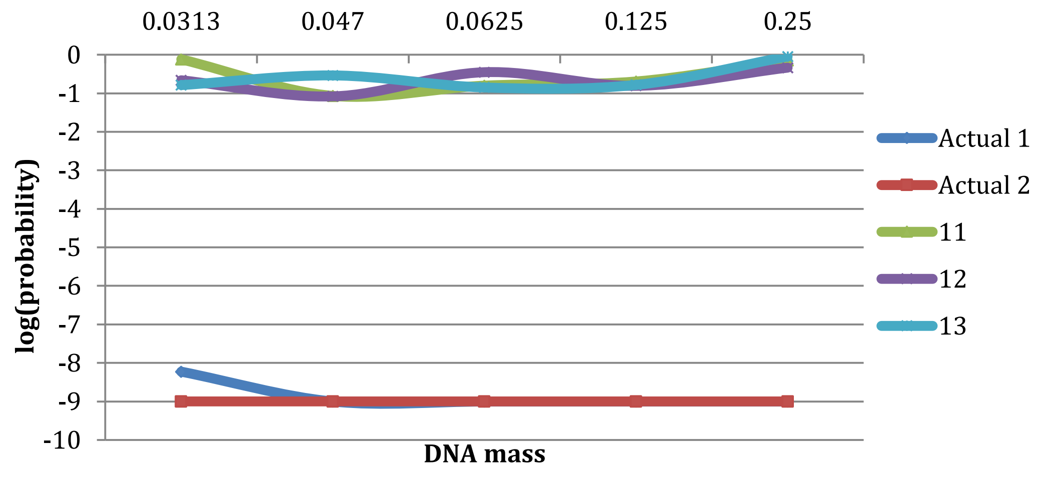


Figure 6 Graph showing results for MatchIt for 2p sample

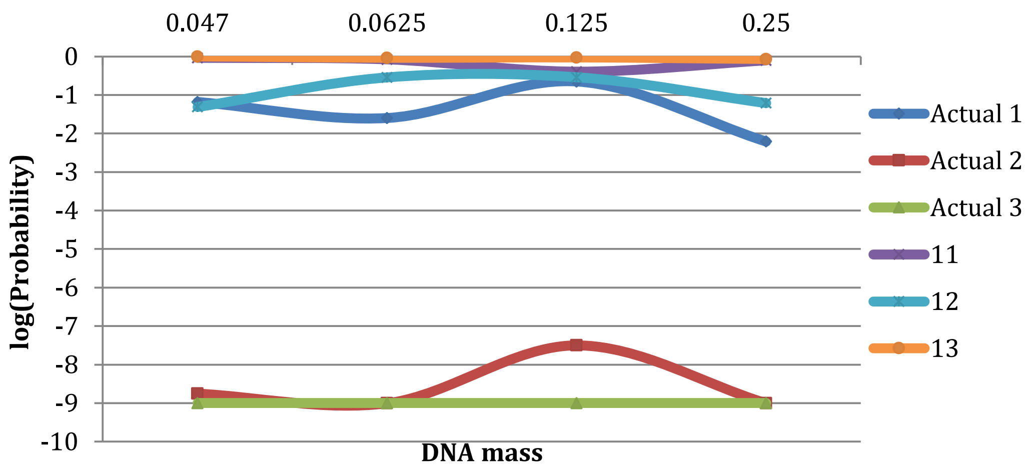


Figure 7 Graph showing results for MatchIt for 3p sample

As the number of contributors increases chances of deflection in the correct also increases. This could also be seen in Figure 7, but the values are clear enough and do not hinder the integrity of the result.

Code Optimization using various techniques

4.1 Introduction

In general, a software could be optimized in three different aspects 1) reduce the running time, 2) use less memory or resources, and 3) draw less power [16]. In this paper our main focus was to reduce the running time of the software. As other two aspects are not that relevant in correspondence to the software. Memory or resources requirements and power usage for this software are as less as to what a personal computer is equipped with these days, which could also be phrased as – 2.3 GHz central processing unit, 1GHz RAM. Software requirements are - windows 7 or later, macintosh X or later, java version 8. There is a widely accepted "rule of thumb"[17] in speed optimization known as the Pareto Principle [18] phrasing that almost 90% of the execution time is spend executing only 10% of the code.

An optimization technique known as "Profile-guided optimization technique" [19] has been used to find the part of code, which is consuming maximum running time. This optimization technique is based on profiling the code, where a dynamic program analysis is done to measure the space (memory), running time of a program, or frequency and duration of function calls [20]. Software was run using a code profiler called "JProfiler" to analyze the running time (CPU cycles) of each class, routine, and line of code during the execution of the software. Then, various optimization techniques (mentioned in Appendix_A) were applied to the code, prioritizing the functions that had longer running times. Changes were accepted if they resulted in an improvement in the runtime performance and discarded otherwise.

4.2 Example of one code snippet

The following function was called around 100,000 times in the software in one execution.

```
public double[] calcSlopeValue(String locusName,double massValue,
HashMap<String,double[]> meanSlope, HashMap<String,double[]> stdDevSlope) {
        try {
                double a_mean = meanSlope.get(locusName)[0];
                double b_mean = meanSlope.get(locusName)[1];
                double a_stddev = stdDevSlope.get(locusName)[0];
                double b_stddev = stdDevSlope.get(locusName)[1];
                double mean = (a_mean * massValue)+b_mean;
                double stddev = (a_stddev * massValue)+b_stddev;
                return new double[] {mean, stddev};
        } catch (Exception e) {
              return null;
        }
}
```

We changed the input parameters from HashMaps to Arrays, which saved approximately 2% of the total running time (CPU cycles). Code became easy to understand and simple to use. Modified code snippet is as follows:

```
public double[] calcSlopeValue(double[] aMean,double[] aStdDev, double massValue) {

        try {

                double mean = (aMean[0] * massValue)+aMean[1];

                double stddev = (aStdDev[0] * massValue)+aStdDev[1];

                return new double[] {mean, stddev};

                } catch (Exception e) {

                        return null;

                }

        }
```

4.3 Results

Created Allele class which saved 70% of auto-boxing conversions from primitive data type to Wrapper class. After this implementation, saved 21.5% in running time.

Converted few HashMaps(get function) to Arrays which reduced overall running time by 14.2%.

Converted few HashMaps(put function, indexing) to ArrayLists which reduced overall running time by 7%.

Changed the implementation of two base methods (calcTwoExpValue and calcSlopeValue and calcExpValue), which had HashMap parameters. This technique implementation saved 10% of the overall running time.

Histogram shown in Figure 8 shows the reduction in time for the core calculation of the program after each optimization technique mentioned in the Table 1. There is an approximately 40% improvement in the overall running time for the core calculation.

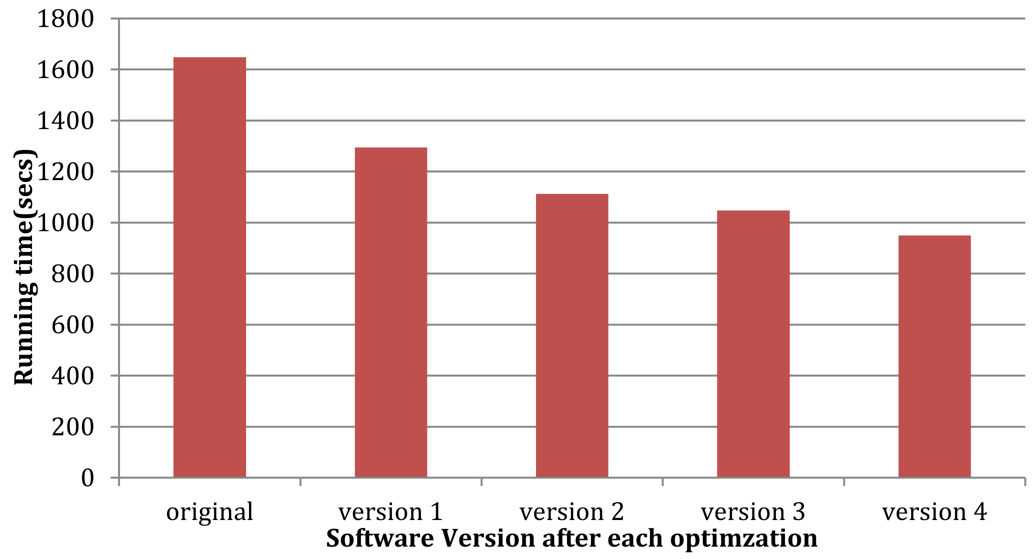


Figure 8 Reduction in running time of base code snippet with each optimization

Graph shown in Figure 9 shows the percent gain in the overall run time of the software.

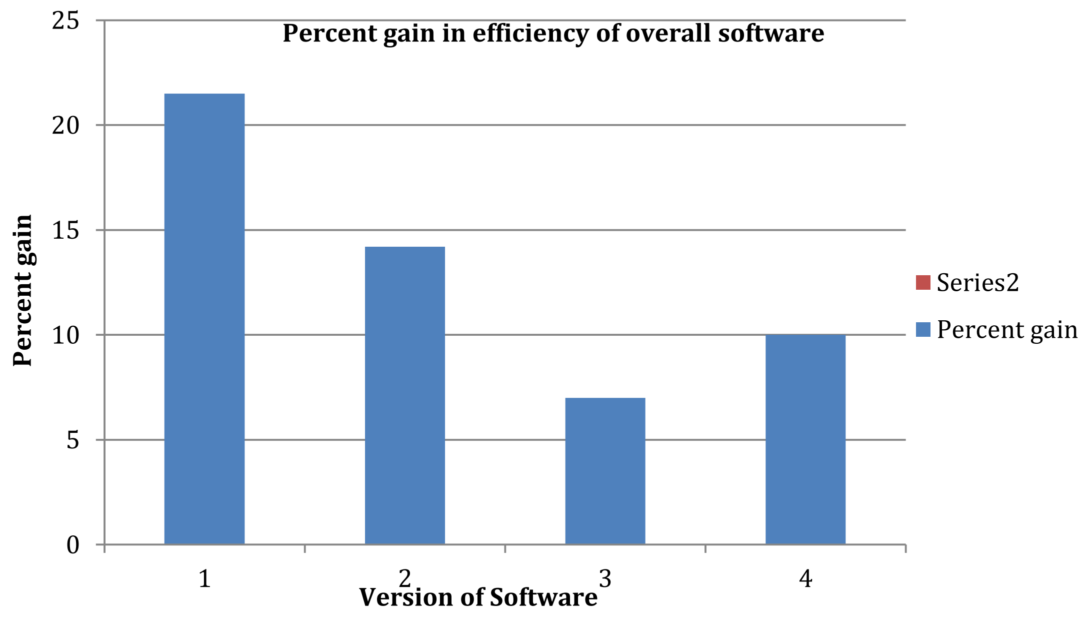


Figure 9 Software efficiency percent gain in different versions

User interface design and development

5.1 Introduction

Having an interface, which makes it easier for the user to handle the software, is an important part of a software package. For example, in calculating curve fitting data during the calibration process (see Section 3.2), it is important to develop a user interface that allows users to easily see the results of the calibration and to modify it interactively. One part of this work was to come up with such an easy-to-use interface for the software, and to display calculated calibration data in a proper manner.

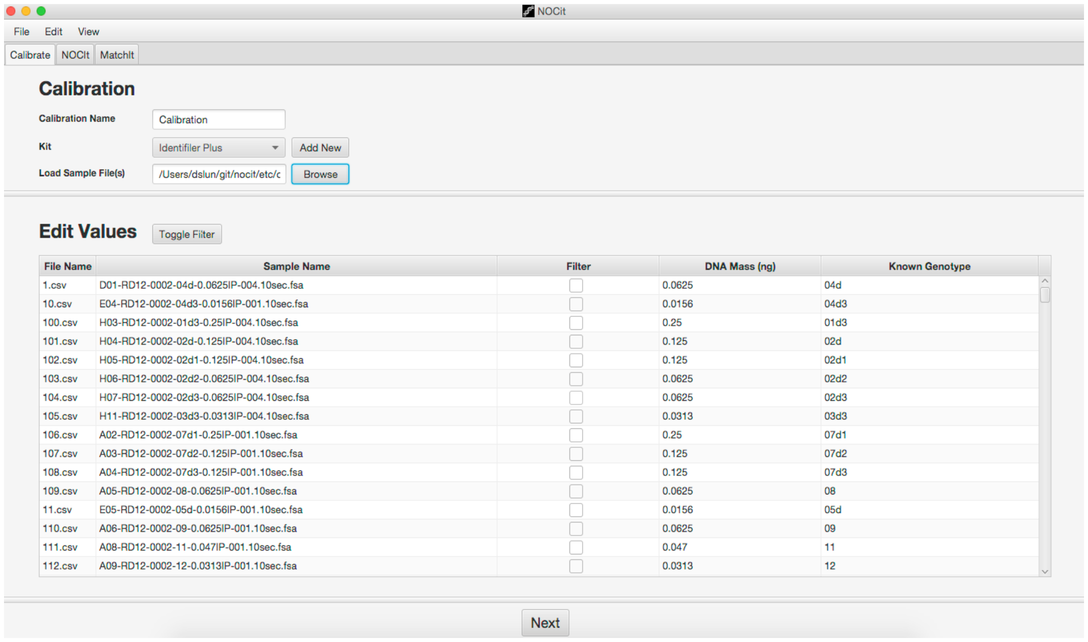
5.2 Design and Implementation

Various versions of the software were developed, paying attention to several principles in user design [21]. The software interface is divided in three different tabs: (1) Calibration, (2) NOCIt, and (3) MatchIt. The calibration tab helps the user to input the data files and parameters. The next window of the calibration tab can be used to input the parameters used for curve fitting. After the calculation, the user can easily navigate using a tree table on the left, and can see the data in an easy to read graphical form on the right side of the window.

Similarly, for the NOCIt and MatchIt tabs, the user can input parameters for the calculation and select the corresponding calibration data generated in the previous step. Based on the parameters, the result is displayed in an easy to understand graphical form, which is displayed in the lower window of the respective tab.

5.3 Results

Calibration is the tab from which our software starts with, as it is usually the first step in the whole process. Figure 8 shows start of the software where user can input the parameters and data files required to generate calibration data.



Calibration

Calibration Name: Calibration

Kit: Identifier Plus

Load Sample File(s): /Users/dslun/git/nocit/etc/c

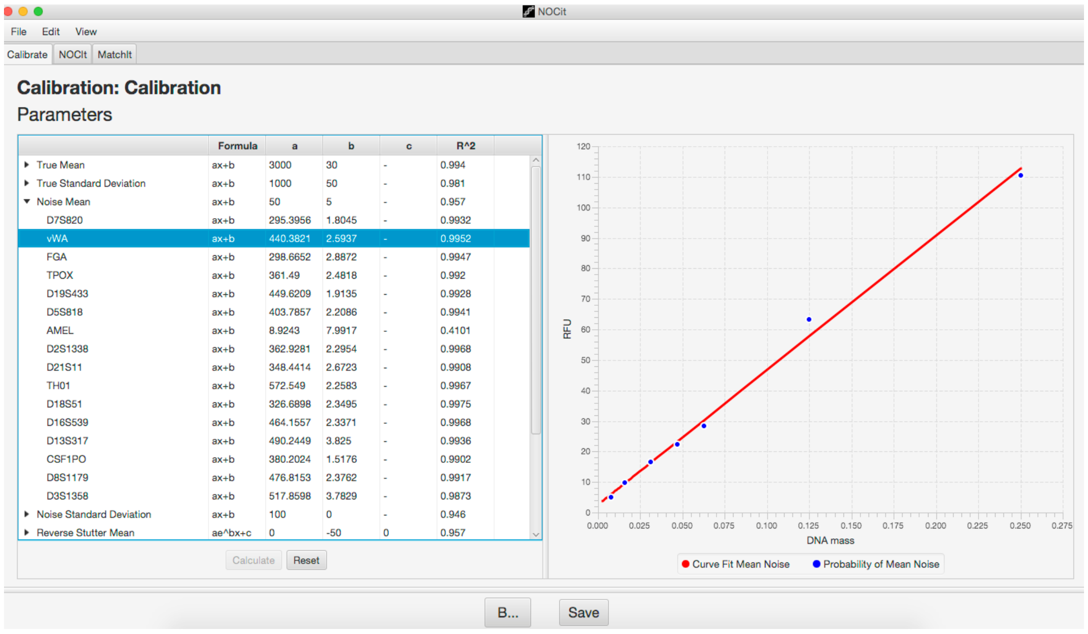
Edit Values

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	E05-RD12-0002-05d-0.0156IP-001.10sec.fsa	☐	0.0156	05d
110.csv	A06-RD12-0002-09-0.0625IP-001.10sec.fsa	☐	0.0625	09
111.csv	A08-RD12-0002-11-0.047IP-001.10sec.fsa	☐	0.047	11
112.csv	A09-RD12-0002-12-0.0313IP-001.10sec.fsa	☐	0.0313	12

Figure 10 First tab of software - Calibration

After inputting the parameters on first window of the calibration tab, user is supposed to click on the "Next" button. This button will take the user to next window of calibration tab as shown below in Figure 9.

User can select parameters and values associated with each of the marked fields in the table and then have to click on the "Calibrate" button to start the calculation. After a few seconds, calculated calibration data will be displayed in a tabular form. User can select any row from the displayed tree table which will display the corresponding graph on the right.



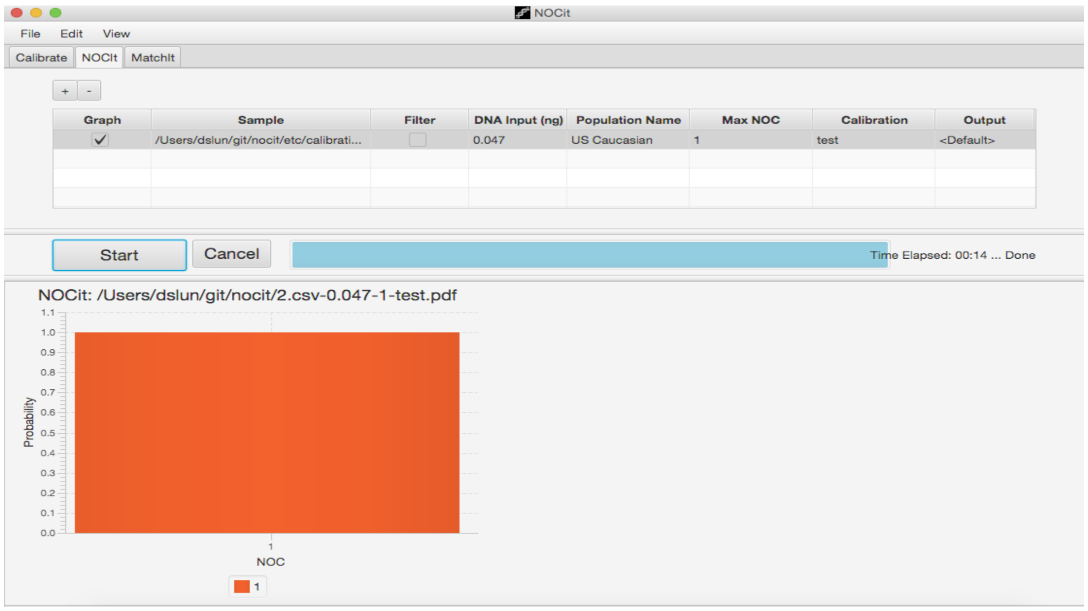
Calibration: Calibration

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
D7S820	ax+b	295.3956	1.8045	-	0.9932
vWA	ax+b	440.3821	2.5937	-	0.9952
FGA	ax+b	298.6652	2.8872	-	0.9947
TPOX	ax+b	361.49	2.4818	-	0.992
D19S433	ax+b	449.6209	1.9135	-	0.9928
D5S818	ax+b	403.7857	2.2086	-	0.9941
AMEL	ax+b	8.9243	7.9917	-	0.4101
D2S1338	ax+b	362.9281	2.2954	-	0.9968
D21S11	ax+b	348.4414	2.6723	-	0.9908
TH01	ax+b	572.549	2.2583	-	0.9967
D18S51	ax+b	326.6898	2.3495	-	0.9975
D16S539	ax+b	464.1557	2.3371	-	0.9968
D13S317	ax+b	490.2449	3.825	-	0.9936
CSF1PO	ax+b	380.2024	1.5176	-	0.9902
D8S1179	ax+b	476.8153	2.3762	-	0.9917
D3S1358	ax+b	517.8598	3.7829	-	0.9873
Noise Standard Deviation	ax+b	100	0	-	0.946
Reverse Stutter Mean	ae^bx+c	0	-50	0	0.957

Figure 11 Second window of Calibration tab with the resuling data

Once the calibration data is generated, user can either save it or can directly go to any of the next two tabs NOCIt and MatchIt. Below Figure 10 shows the NOCIt tab of the software where user can select any number of rows, each with the appropriate parameters to start NOCIt calculation. Result will be shown as a histogram on the bottom of the window.



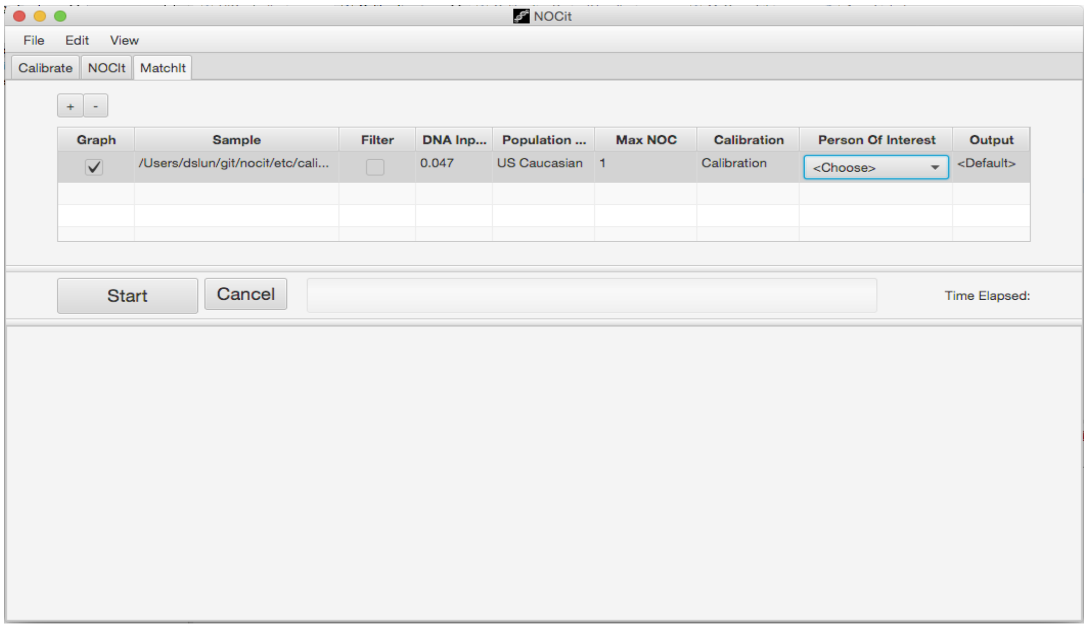
Graph	Sample	Filter	DNA Input (ng)	Population Name	Max NOC	Calibration	Output
✓	/Users/dslun/git/nocit/etc/calibrati...		0.047	US Caucasian	1	test	<Default>

Time Elapsed: 00:14 ... Done

NOCit: /Users/dslun/git/nocit/2.csv-0.047-1-test.pdf

Figure 12 NOCit tab with the result

MatchIt tab is very similar to NOCIt tab where user can select various rows with appropriate parameters and will get the result in the form of a histogram on the bottom of the window. Figure 11 shows the MatchIt tab without a result.



Graph	Sample	Filter	DNA Inp...	Population ...	Max NOC	Calibration	Person Of Interest	Output
✓	/Users/dslun/git/nocit/etc/cali...	☐	0.047	US Caucasian	1	Calibration	<Choose>	<Default>

Figure 13 MatchIt tab

Appendix A

Optimization Technique	Example	Reason and Improvement

Common sub-expression elimination	a = b * c + g;
	d = b * c * e;
	To
	tmp = b * c;
	a = tmp + g;
	d = tmp * e;
		Saves various operations

Code motion	for(int x : array)
	    value += func(3) * x;
	To
	tmp = func(3);
	for(int x : array)
	    value += tmp * x;
		Saves multiple function calls

Use arrays instead of HashMaps	HashMap<Integer,Double> valueMap = new HashMap<>();
	To
	double[] valueMap = new double[size];
		Saves conversion from primitive data-type to corresponding object. Overall calculation becomes much more efficient.

Unrolling loops	for (int j=0; j< 2-l; j++) {
	    s = "0"+s;
	  }
	TO
	for (int j=0; j< 2-l; j+=2) {
		Saves multiple integer operations

	s = "0"+s;
	
	s = "0"+s;
	
	}	

Removal of dead or unreachable code	Helps in decreasing memory utilization	Saves unnecessary memory loads
Constant folding	x = 2.0 * x * 4.0 to x=8 * x	Saves a floating point operation

Loop invariant optimization	While x>0
	
	x = x – (y+z);
	
	to
	
	t = y+z;
	
	while (x>0)
	
	x = x – t;
		Saves multiple integer addition operations

Dead-variable elimination	Checked, analyzed and removed unnecessary dead variables	Saves a memory operation

Changed LinkedinHashMap<Integer,Integer> to ArrayList<Integer>	LinkedinHashMap<Integer,Integer> value = new LinkedinHashMap<>();
	
	To
	
	ArrayList<Integer> value = new ArrayList<>();
		Saves conversion from primitive data-type to corresponding object. Overall calculation becomes much more efficient.

Changed ArrayLists to Arrays	ArrayList<Integer> value = new ArrayList<>();
	
	To
	
	int[] value = new int[size];
		Saves conversion from primitive data-type to corresponding object. Overall calculations becomes much more efficient.

REFERENCES

1. J.M. Butler, Fundamentals of forensic DNA typing, first ed., Associated Press, 2009.

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

3. J. Buckleton, J. Curran, A discussion of the merits of random man not excluded and likelihood ratios, Forensic Sci Int Genet. 2 (2008) 343-348. DOI: 10.1016/j.fsigen.2008.05.005.

4. C.M. Grgicak, Z.M. Urban, R.W. Cotton, Investigation of Reproducibility and Error Associated with qPCR Methods using Quantifiler Duo DNA Quantitation Kit. J Forensic Sci. 55 (2010) 1331-1339. DOI: 10.1111/j.1556-4029.2010.01460.x.

5. M.C. Cicero, C.M. Grgicak, Examination into the Applicability and Stability of a Single External Calibrator for Forensic DNA Quantification. NorthEastern Association of Forensic Scientists, Cromwell, CT, 2013.

6. Applied Biosystems, AmpFlstr® Identifiler® Plus PCR Amplification Kit Users' Manual, first ed., 2006.

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

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

9. M.W.Perlin, K.Dormer, J.Hornyak, L. Schiemer-Wood, S. Greenspoon, TrueAllele Casework on Virginia DNA Mixture EvidenceL Comnputer and Manual Interpretation in 72 Reported Criminal Cases. PLoS ONE 9 (2014). DOI: 10.1371/journal.pone.0092837.

10. R.G. Cowell, S.L. Lauritzen, J. Mortera, Probabilistic expert systems for handling artifacts in complex DNA mixtures, Forensic Sci Int Genet. 5 (2011) 202-209. DOI: 10.1016/j.fsigen.2010.03.008.

11. R. Puch-Solis, L. Rodgers, A. Mazumder, S. Pope, I. Evett, J. Curran, D. Balding, Evaluating forensic DNA profiles using peak heights, allowing for multiple donors, allelic dropout and stutters, Forensic Sci Int Genet. 7 (2013) 555-563. DOI: 10.1016/j.fsigen.2013.05.009.

12. D. Taylor, J. Bright, J. Buckleton, The interpretation of single source and mixed DNA profiles, Forensic Sci Int Genet. 7 (2013) 516-528. DOI: 10.1016/j.fsigen.2013.05.011.

13. D.J.C. Mackay, Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003.

14. J.M. Butler, Fundamentals of forensic DNA typing, first ed., Associated Press, 2009.

15. D.J.C. Mackay, Introduction to Monte Carlo Methods, Springer Netherlands, 1998.

16. Optimizing compiler. (n.d.). In Wikipedia. Retrieved March, 2015, from https://en.wikipedia.org/wiki/Optimizing_compiler

17. Rule of Thumb, (n.d.). In Wikipedia. Retrieved March, 2015, from https://en.wikipedia.org/wiki/Rule_of_thumb

18. Pareto principle, (n.d.). In Wikipedia. Retrieved March, 2015, from https://en.wikipedia.org/wiki/Pareto_principle

19. Profile-guided optimization, (n.d.). In Wikipedia. Retrieved March, 2015, from https://en.wikipedia.org/wiki/Profile-guided_optimization

20. Profiling (Computer programming), (n.d.). In Wikipedia. Retrieved March, 2015, from https://en.wikipedia.org/wiki/Profiling_(computer_programming)

21. Principles of user interface design, (n.d.). In Wikipedia. Retrieved March, 2015, from https://en.wikipedia.org/wiki/Principles_of_user_interface_design