

A tutorial for the package *forensim*

Hinda Haned

July 23, 2009

Contents

1	Introduction	3
2	Getting started	3
2.1	forensim installation	3
2.2	How to get help	3
3	Generating data in forensim	4
3.1	tabfreq objects	4
3.2	simugeno objects	6
3.3	simumix objects	7
3.4	Allele frequencies simulation	8
3.4.1	The homogeneous population case	8
3.4.2	The subdivided population case	8
4	Statistical methods for forensic DNA mixtures interpretation	9
4.1	The maximum allele count	10
4.2	The maximum likelihood estimator	11
4.2.1	Likelihood of the observed alleles at a given locus, conditional on the number of contributors to the mixture	11
4.2.2	Maximum likelihood estimators	12
4.3	The exclusion probability	12
4.4	The random match probability	13
4.5	Likelihood ratios	14
5	Two-person DNA mixtures resolution using allele peak heights or areas information: The <i>mastermix</i> interface	15
6	Miscellaneous	19
6.1	Manipulating forensim objects	19
6.2	How to change population names	20
6.3	How to find the allele frequencies of a mixture	20
6.4	The number of alleles in a mixture	21
	References	22

A	Appendix: Formulas used in <i>mastermix</i>	24
A.1	Expected allelic ratios	24
A.2	Conditional mixtures proportions	26

1 Introduction

This tutorial is a presentation of the `forensim` package for the R software [1, 2]. `forensim` is dedicated to the interpretation of forensic DNA mixtures through statistical methods. It also provides simulation tools that allow the generation of genetic data commonly encountered in forensic casework.

In this tutorial, I first introduce `forensim` object classes. Then, I present statistical tools for forensic DNA mixtures interpretation. Finally, various functionalities of `forensim` are explored. For all addressed topics, practical and reproducible examples are given.

2 Getting started

2.1 `forensim` installation

Latest stable version. To be sure to get the latest stable version, download the `forensim` package (according to your platform) on `forensim` web page: <http://forensim.r-forge.r-project.org/>. The current version of the package is 1.1-1 and is compatible with R 2.9.1.

The latest developpement build. `forensim` is hosted by Rforge, the latest developpement build of the package, resulting from the nightly build, can be obtained by typing the following command lines:

- Under Windows and Linux

```
> install.packages("forensim", repos="http://r-forge.r-project.org")
```

- Under the MacOS system

```
> install.packages("forensim", repos="http://r-forge.r-project.org", type = 'source')
```

Please be aware that this is the development version, so some errors might occur. Once the package is downloaded to your system, it must be loaded:

```
> library(forensim)
```

```
### forensim 1.1.1 is loaded ###
```

2.2 How to get help

- The mailing list: please ask questions on `forensim` mailing list, forensim-help@lists.r-forge.r-project.org
- The help pages: classes and functions are documented in the help pages, type `?forensim` in R to get an overview of the package.
- The `forensim` package manual: a compilation of all the help pages in a single pdf file, it can be found at: <http://forensim.r-forge.r-project.org/>

3 Generating data in `forensim`

`forensim` provides object classes that facilitate the generation and the storage of data that is commonly encountered in forensic casework: population allele frequencies, individual genotypes and DNA mixtures. Thus, three classes of objects are defined in `forensim`:

- `tabfreq` objects: used to store allele frequencies
- `simugeno` objects: used to store genotypes
- `simumix` objects: used to store DNA mixtures

`forensim` objects have the particularity that they can either be used to store pre-existing data, such as allele frequencies in a given population, or simulated data. Creating `forensim` objects is achieved using specific functions, called constructors, that have the same names than the object they are linked to.

3.1 `tabfreq` objects

In `forensim`, allele frequencies are stored in `tabfreq` objects. Importing data into `tabfreq` objects is achieved using the `tabfreq` constructor. The input data must be an object of type data frame¹ or matrix. This object must have the format of the *Journal of Forensic Sciences* for Short Tandem Repeat (STR) loci data: allele names (the number of tandem repeats in case of STR loci) are given in the first column, and frequencies for a given allele are read in rows for different loci given in columns. When an allele is not observed for a given locus, value is coded “NA”². Note that even if the requested input format is based on STR data, different kinds of markers can be imported in `forensim`.

As an example, we will be using a data set included in `forensim`:

```
> data(Tu)
```

What is the class of object `Tu` ?

```
> class(Tu)
```

```
[1] "data.frame"
```

`Tu` is a data frame giving the allele frequencies for 15 STR loci commonly used in forensic studies, in the Tu Chinese population [3] (see `?Tu`). Note that the data set is imported using the command `data`.

Displaying the first rows (command `head`):

```
> head(Tu)
```

¹in R a data frame is a collection of variables, possibly of different types

²non observed alleles are coded “-” in the *Journal of Forensic Sciences*

	Allele	D8S1179	D21S11	D7S820	CSF1P0	D3S1358	TH01	D13S317	D16S539	D2S1338
1	6.0	NA	NA	NA	NA	NA	0.1151	NA	NA	NA
2	7.0	NA	NA	0.0033	0.0034	NA	0.2599	NA	NA	NA
3	8.0	0.0098	NA	0.1382	0.0034	NA	0.0559	0.2712	0.0097	NA
4	9.0	NA	NA	0.0493	0.0582	NA	0.4605	0.1503	0.2305	NA
5	9.2	NA	NA	0.0033	NA	NA	NA	NA	NA	NA
6	9.3	NA	NA	NA	NA	NA	0.0691	NA	NA	NA

	DS19S433	vWA	TPOX	D18S51	D5S818	FGA
1	NA	NA	NA	NA	NA	NA
2	NA	NA	NA	NA	0.0097	NA
3	NA	NA	0.5359	NA	NA	NA
4	NA	NA	0.1340	NA	0.0487	NA
5	NA	NA	NA	NA	NA	NA
6	NA	NA	NA	NA	NA	NA

This data frame is converted into a `tabfreq` object by the `tabfreq` constructor:

```
> tupop <- tabfreq(tab = Tu, pop.names = as.factor("Tu"))
```

The population name is specified as a factor in the `pop.names` argument.

```
> is.tabfreq(tupop)
```

```
[1] TRUE
```

tupop is a tabfreq object:

```
> tupop
# Tabfreq object: allele frequencies #
```

```
@tab: list of allele frequencies
@which.loc: vector of 15 locus names
@pop.names: populations names
```

As a formal class object, `tabfreq` is constituted of different 'slots' that contain different types of information. Each slot can be accessed using '@' or the '\$' operator that have been implemented for all `forensim` objects.

Allele frequencies are stored in the `@tab` slot. For example, frequencies for locus FGA are given by:

```
> tupop$tab$Tu$FGA
      18      19      19.2      20      21      22      22.2      23      23.2      24      25
0.0392 0.0686 0.0033 0.0458 0.0980 0.1765 0.0033 0.1961 0.0098 0.2222 0.1013
      25.2      26      26.2      27
0.0065 0.0131 0.0065 0.0098
```

Population names are stored in the `@pop.names` argument:

```
> tupop$pop.names
```

```
[1] Tu
Levels: Tu
```

Finally, locus names appearing in `@tab` can be accessed elsewhere:

```
> tupop$which.loc
[1] "D8S1179" "D21S11" "D7S820" "CSF1P0" "D3S1358" "TH01"
[7] "D13S317" "D16S539" "D2S1338" "DS19S433" "vWA" "TPOX"
[13] "D18S51" "D5S818" "FGA"
```

Note that if several populations are imported in the same `tabfreq` object, data frames (or matrices) must be given as a list of data frames (or matrices) in the `tab` argument. In this case, the `pop.names` argument, which is optional when a single population is handled, becomes obligatory in order to distinguish the populations.

3.2 simugeno objects

`simugeno` objects are used to store simulated genotypes from a `tabfreq` object. `simugeno` objects are created from `tabfreq` objects by specifying the number of individuals to simulate in the `n` argument. The loci to take into account for the simulation are given in the `which.loc` argument. For the illustration purpose, 10 individuals are simulated and only three loci are chosen: D8S1179, TH01 and FGA.

```
> tugeno <- simugeno(tab = tupop, n = 10, which.loc = c("D8S1179",
+ "TH01", "FGA"))
```

```
> tugeno
```

```
# Simugeno object: simulated genotypes #

@which.loc: vector of 3 locus names
@nind: 10
@indID: vector of the individuals ID
@tab.geno: 10 x 3 data frame of genotypes
@tab.freq: allele frequencies for the 3 loci

Population-related information:
@pop.names: population names
@popind: factor giving the population of each individual
```

`@tab.geno` is a matrix of 10 genotypes simulated from the allele frequencies of the Tu population. For instance, the genotypes of the five first simulated individuals are:

```
> tugeno$tab.geno[1:5, ]

      D8S1179 TH01   FGA
ind1 "10/14" "7/9"  "23/21"
ind2 "14/13" "9/9"  "20/25"
ind3 "15/16" "8/7"  "23/21"
ind4 "8/14"  "9/9.3" "22/18"
ind5 "13/12" "7/7"  "23/21"
```

The genotype of a homozygous individual carrying the allele 9 is coded "9/9". A heterozygous individual carrying alleles 8 and 10 is coded "8/10".

Allele frequencies of the population are stored in the slot `@tab.freq`:

```
> tugeno$tab.freq

$Tu
$Tu$D8S1179
      8      10      11      12      13      14      15      16      17
0.0098 0.0784 0.0784 0.1046 0.2876 0.1863 0.1634 0.0719 0.0196

$Tu$TH01
      6      7      8      9      9.3      10
0.1151 0.2599 0.0559 0.4605 0.0691 0.0395

$Tu$FGA
      18      19      19.2      20      21      22      22.2      23      23.2      24      25
0.0392 0.0686 0.0033 0.0458 0.0980 0.1765 0.0033 0.1961 0.0098 0.2222 0.1013
      25.2      26      26.2      27
0.0065 0.0131 0.0065 0.0098
```

`simugeno` objects also contain information about the simulated individuals, their (default) ID:

```
> tugeno@indID
```

```
[1] "ind1" "ind2" "ind3" "ind4" "ind5" "ind6" "ind7" "ind8" "ind9"
[10] "ind10"
```

and their population names:

```
> tugeno@popind
```

```
[1] Tu Tu Tu Tu Tu Tu Tu Tu Tu Tu Tu
Levels: Tu
```

3.3 simumix objects

`simumix` objects store DNA mixtures. Mixtures can be created from `simugeno` objects using the constructor `simumix`. The number of contributors is specified in the argument `ncontri`.

```
> mix2 <- simumix(tugeno, ncontri = 2)
```

Constructor `simumix` has also a `which.loc` argument, which is by default set to `NULL`, corresponding to all loci taken into account.

```
> mix2
```

```
# Simumix object: simulated mixture #
```

```
@which.loc: vector of 3 locus names
@ncontri: 2
@mix.prof: 2 x 3 data frame of the contributors genotypes
@mix.all: list of the alleles found in the mixture
@popinfo: populations of the contributors
```

`simumix` objects keep two types of information: information usually available when dealing with practical cases of forensic DNA mixtures: the alleles present by locus,

```
> mix2$mix.all
```

```
$D8S1179
[1] "11" "15" "16"
```

```
$TH01
[1] "7" "8" "9" "9.3"
```

```
$FGA
[1] "19" "21" "23" "24"
```

and information that is usually not available: the number of simulated contributors

```
> mix2@ncontri
```

```
[1] 2
```

and their genetic profiles:

```
> mix2$mix.prof
```

```
      D8S1179 TH01   FGA
ind3 "15/16" "8/7" "23/21"
ind8 "16/11" "9.3/9" "24/19"
```

3.4 Allele frequencies simulation

In the following, we denote L a locus with k alleles and the i th allele frequency at this locus, in a given population, is denoted p_i .

3.4.1 The homogeneous population case

In *forensim*, allele frequencies for a single non subdivided population are simulated using the `simufreqD` function.

Principle

The vector of allele frequencies at locus L is simulated as a vector of random deviates of the Dirichlet distribution [4] with a vector of parameters $(\alpha_1, \dots, \alpha_k)$:

$$(p_1, \dots, p_k) \rightsquigarrow \text{Dirichlet}(\alpha_1, \dots, \alpha_k)$$

An example

5 loci (argument `nloc=5`) having 2, 3, 4, 5 and 6 alleles respectively (argument `na`) are simulated:

```
> simufreqD(nloc = 5, na = c(2, 3, 4, 5, 6), alpha = 1)
```

	Allele	Marker1	Marker2	Marker3	Marker4	Marker5
1	1	0.6392141	0.41940450	0.0902012	0.01416308	0.015178548
2	2	0.3607859	0.55596167	0.2801002	0.31777097	0.465132752
3	3	NA	0.02463383	0.2271227	0.24096551	0.006043277
4	4	NA	NA	0.4025759	0.34643984	0.008724610
5	5	NA	NA	NA	0.08066061	0.083311472
6	6	NA	NA	NA	NA	0.421609342

Argument `alpha` is the parameter of the Dirichlet distribution. Setting a single value for `alpha` means that all alleles for all loci are simulated with the same value; this can be changed by giving the appropriate values in `alpha`, for further details please type `'?simufreqD'`.

Setting `alpha` to 1, leads to the generation of allele frequencies as random deviates from a uniform Dirichlet distribution, this means that allele frequencies could take any value varying from 0 to 1, with equal probabilities. Note that the simulated data is in the format of the *Journal of Forensic Sciences* for STR loci data.

3.4.2 The subdivided population case

Principle

The `simupopD` function simulates subpopulations allele frequencies for independent loci, from a given reference population, following a Dirichlet model. Allele frequencies in the subpopulations are generated as random deviates from a Dirichlet distribution, the parameters of which control the deviation of allele frequencies from the values in the reference population.

Each allele frequency is modeled as a random variable; with a parameter

$\alpha_i = \frac{p_i(1 - \theta)}{\theta}$, where θ is Wright's *Fst* coefficient which allows here accounting for population subdivision [5, 6]. The vector of allele frequencies at a given locus, for a

given population, is obtained by:

$$(p_1, \dots, p_k) \rightsquigarrow \text{Dirichlet} \left(\alpha_1 = \frac{p_1(1 - \theta)}{\theta}, \dots, \alpha_k = \frac{p_k(1 - \theta)}{\theta} \right)$$

An example

In the following example we simulate allele frequencies in two subpopulations: the global population is taken as the Tu Chinese population, and three STR loci are chosen: FGA, TH01 and TPOX. The strength of the deviation from the reference allele frequencies is specified in argument `alpha1` for each simulated subpopulation, here we choose 0.01 for the first population and 0.3 for the second one:

```
> simpop1 <- simupopD(npop = 2, globalfreq = Tu, which.loc = c("FGA",  
+ "TH01", "TPOX"), alpha1 = c(0.01, 0.3))
```

`simpop1` is a list of two `tabfreq` object; the first one contains allele frequencies used for the simulation (from the Tu population):

```
> simpop1$globfreq  
  
# Tabfreq object: allele frequencies #  
  
@tab: list of allele frequencies  
@which.loc: vector of 3 locus names  
@pop.names: - empty -
```

the second `tabfreq` object contains the subpopulations allele frequencies:

```
> simpop1$popfreq  
  
# Tabfreq object: allele frequencies #  
  
@tab: list of allele frequencies  
@which.loc: vector of 3 locus names  
@pop.names: populations names
```

The simulated subpopulations have the following (default) names:

```
> simpop1$popfreq$pop.names  
  
[1] pop1 pop2  
Levels: pop1 pop2
```

4 Statistical methods for forensic DNA mixtures interpretation

Several statistical methods dedicated to the interpretation of forensic DNA mixtures are implemented in `forensim`:

4.1 The maximum allele count

This method consists in setting the lower bound on the number of contributors to a mixture to the minimum required to explain the observed profiles [7]. For instance, if a mixture shows at three loci, 1, 3 and 4 alleles, then the number of contributors is bounded to $2 \left(\frac{4}{2} \right)$ contributors.

To exemplify this method, let us simulate a 3-person mixture from the `strusa` data set, using the allele frequencies from the Caucasian population [8] (see `?strusa`):

```
> data(strusa)
> class(strusa)

[1] "tabfreq"
attr(,"package")
[1] ".GlobalEnv"

> strusa

# Tabfreq object: allele frequencies #

@tab: list of allele frequencies
@which.loc: vector of 15 locus names
@pop.names: populations names
```

`strusa` is a `tabfreq` object that contains multiple populations:

```
> strusa$pop.names

[1] Afri Cauc Hisp
Levels: Afri Cauc Hisp
```

thus, the number of genotypes to simulate must be specified in each population (argument `n`):

```
> geno <- simugeno(tab = strusa, n = c(0, 100, 0))
```

100 genotypes are simulated from the Caucasian population allele frequencies, no genotypes are simulated from the other two populations.

A 3-person mixture is simulated by randomly drawing three contributors from these 100 simulated individuals. The number of contributors in each population must be specified:

```
> mix3 <- simumix(tab = geno, ncontri = c(0, 3, 0))
```

The minimum number of contributors required is computed by the `mincontri` function. This number can either be computed from all available loci simultaneously (in this default case, the argument `loc` is set to `NULL`),

```
> mincontri(mix3, loc = NULL)
```

```
[1] 3
```

or be computed for a specific locus, for example, D8S1179:

```
> mincontri(mix3, loc = "D8S1179")
```

```
[1] 2
```

4.2 The maximum likelihood estimator

The main characteristic of this method is that it takes into account allele frequencies in the estimations. The likelihood function is derived from the formula of Curran *et al* [9] for DNA mixtures interpretation, in the particular case where all contributors to the mixture are unknown and there are no typed individuals [10].

4.2.1 Likelihood of the observed alleles at a given locus, conditional on the number of contributors to the mixture

The function `lik.loc` computes the likelihood of the observed alleles at a given locus, conditional on the number of contributors to the mixture [10]. This function takes in argument the number of contributors `x`, the mixture as a `simumix` object, and the allele frequencies given in a `tabfreq` object. For the previously simulated 3-person mixture `mix3`,

```
> mix3

# Simumix object: simulated mixture #

@which.loc: vector of 15 locus names
@ncontri: 3
@mixture: 3 x 15 data frame of the contributors genotypes
@alleles: list of the alleles found in the mixture
@popinfo: populations of the contributors
```

the likelihood per locus of observing alleles given that 1 individual contributed to the mixture is:

```
> lik.loc(x = 1, mix = mix3, freq = strusa, refpop = "Cauc")

      CSF1PO      FGA      TH01      TPOX      VWA D3S1358 D5S818 D7S820 D8S1179 D13S317
0      0      0      0      0      0      0      0      0      0      0
D16S539 D18S51 D21S11 D2S1338 D19S433
0      0      0      0      0
```

the likelihood that 3 individuals contributed to the mixture is:

```
> lik.loc(x = 3, mix = mix3, freq = strusa, refpop = "Cauc")

      CSF1PO      FGA      TH01      TPOX      VWA      D3S1358
0.3238078841 0.0049592829 0.1495254898 0.0957964185 0.0773829861 0.0081035646
      D5S818      D7S820      D8S1179      D13S317      D16S539      D18S51
0.0748500868 0.0042380878 0.0623661401 0.0275685321 0.0440544002 0.0039501213
      D21S11      D2S1338      D19S433
0.0004864817 0.0003866552 0.0012297968
```

Note here that `strusa` contains three populations, so the reference population, here Caucasians, must be specified in the `refpop` argument.

The overall likelihood, for all loci characterized in the mixture can be computed using the function `lik`:

```
> lik(x = 3, mix = mix3, freq = strusa, refpop = "Cauc")
```

```
[1] 3.166945e-28
```

4.2.2 Maximum likelihood estimators

`likestim.loc` looks for the number of contributors that maximizes the likelihood at each given locus. For the estimations to be biologically plausible, the estimations are restricted to the discrete interval [1,6] [10]. These functions give the number of contributors that maximizes the likelihood (max) and the corresponding likelihood value (maxval). The per locus estimations are:

```
> likestim.loc(mix = mix3, freq = strusa, reipop = "Cauc")

      CSF1PO      FGA      TH01      TPOX      VWA      D3S1358
max      3.0000000 4.000000000 5.0000000 3.00000000 4.00000000 6.00000000
maxval 0.3238079 0.006355305 0.2247431 0.09579642 0.08159628 0.02969024
      D5S818      D7S820      D8S1179      D13S317      D16S539      D18S51
max      3.00000000 3.000000000 4.00000000 4.00000000 2.00000000 2.00000000
maxval 0.07485009 0.004238088 0.06259472 0.02793542 0.04979156 0.004292059
      D21S11      D2S1338      D19S433
max      3.000000000 4.000000000 3.000000000
maxval 0.0004864817 0.001095117 0.001229797
```

and the estimation using all loci simultaneously is:

```
> likestim(mix = mix3, freq = strusa, reipop = "Cauc")

      [,1]
max      3.000000e+00
maxval 3.166945e-28
```

4.3 The exclusion probability

The exclusion probability, also known as the Random Man Not Excluded (RMNE) is implemented in forensim in the function `PE`.

The `PE` function takes a `simumix` object for which to compute the exclusion probability and the allele frequencies given in a `tabfreq` object. If the latter contains several populations, than the reference population must be specified in the `reipop` argument. Implementation of the `PE` function includes the possibility of correcting for deviation from Hardy Weinberg proportions in the population, due to subdivision, using Wright's *Fst* called here theta [11]:

```
> PE(mix3, strusa, reipop = "Cauc", theta = 0, byloc = TRUE)

      CSF1PO      FGA      TH01      TPOX      VWA      D3S1358      D5S818
PE_1 0.2271129 0.5592368 0.1828202 0.3037099 0.3710717 0.2006925 0.368436
      D7S820      D8S1179      D13S317      D16S539      D18S51      D21S11      D2S1338
PE_1 0.7210681 0.4073002 0.4350373 0.5048908 0.774223 0.6849872 0.6206672
      D19S433
PE_1 0.6699383
```

The row `PE_1` stands for the exclusion probability per locus, read in column. The `byloc` argument is a logical indicating whether the exclusion probability should be computed per locus (`byloc=TRUE`) or for all loci (`byloc=FALSE`):

```
> PE(mix = mix3, freq = strusa, reipop = "Cauc", theta = 0, byloc = FALSE)

      PE
0.9999747
```

4.4 The random match probability

The Random Match Probability (RMP) is computed using the `RMP` function which implements the formulas gave by Balding and Nichols [12]. The suspect's profile can either be given directly in R as matrix, or be read from a text file.

DNA evidence as a matrix

```
> data <- matrix(c("CSF1P0", "FGA", "TH01", "TPOX", "VWA", "D3S1358",
+ "D5S818", "D7S820", "D8S1179", "D13S317", "D16S539", "D18S51",
+ "D21S11", "D2S1338", "D19S433", "12/11", "22/19", "6/7",
+ "10/8", "17/18", "18/17", "12/12", "8/8", "13/13", "11/11",
+ "12/10", "14/15", "33.2/32.2", "23/22", "14/14"), nc = 2)
> colnames(data) <- c("locus", "genotype")
> data
```

```
      locus      genotype
[1,] "CSF1P0" "12/11"
[2,] "FGA"    "22/19"
[3,] "TH01"   "6/7"
[4,] "TPOX"   "10/8"
[5,] "VWA"    "17/18"
[6,] "D3S1358" "18/17"
[7,] "D5S818" "12/12"
[8,] "D7S820" "8/8"
[9,] "D8S1179" "13/13"
[10,] "D13S317" "11/11"
[11,] "D16S539" "12/10"
[12,] "D18S51" "14/15"
[13,] "D21S11" "33.2/32.2"
[14,] "D2S1338" "23/22"
[15,] "D19S433" "14/14"
```

The random match probability in the unrelated case (unknown offender and suspect are not related) and in absence of population subdivision ($\theta=0$, default case) is given by ¹:

```
> RMP(suspect = data, freq = strusa, reipop = "Cauc")
```

NOTE: THIS PACKAGE IS NOW OBSOLETE.

The R-Genetics project has developed an set of enhanced genetics packages to replace 'genetics'. Please visit the project homepage at <http://rgenetics.org> for informtion.

```
$RMP.loc
      CSF1P0      FGA      TH01      TPOX      VWA      D3S1358
0.217510855 0.023156498 0.088265632 0.060204407 0.112769764 0.065567667
      D5S818      D7S820      D8S1179      D13S317      D16S539      D18S51
0.147540492 0.022698436 0.092805530 0.115192360 0.036719093 0.043683070
      D21S11      D2S1338      D19S433
0.004473631 0.008952608 0.136316024
```

```
$RMP
[1] 6.204726e-20
```

In the absence of population subdivision, and in the case where the suspect and an unknown offender are for example siblings, the `k` argument must be modified from $k=(1,0,0)$ to $k=c(1/4,1/2,1/4)$:

¹RMP calls many functions from the genetics package, which is now obsolete. So, don't worry if you get a warning message from the genetics package.

```
> RMP(suspect = data, freq = strusa, k = c(1/4, 1/2, 1/4), refpop = "Cauc")
```

```
$RMP.loc
      CSF1P0      FGA      TH01      TPOX      VWA      D3S1358      D5S818      D7S820
0.4699402 0.3236691 0.3776139 0.4128161 0.3986399 0.3582794 0.4789401 0.3310046
      D8S1179      D13S317      D16S539      D18S51      D21S11      D2S1338      D19S433
0.4255214 0.4484981 0.3547923 0.3350108 0.2788509 0.2911457 0.4686840

$RMP
[1] 4.63387e-07
```

DNA evidence read from an existing text file The same data is available in a preexisting file “exprofile.txt” from the forensim package, accessed by the system.file command:

```
> RMP(filename = system.file("files/exprofile.txt", package = "forensim"),
+      freq = strusa, refpop = "Cauc")
```

```
$RMP.loc
      CSF1P0      FGA      TH01      TPOX      VWA      D3S1358
0.217510855 0.023156498 0.088265632 0.060204407 0.112769764 0.065567667
      D5S818      D7S820      D8S1179      D13S317      D16S539      D18S51
0.147540492 0.022698436 0.092805530 0.115192360 0.036719093 0.043683070
      D21S11      D2S1338      D19S433
0.004473631 0.008952608 0.136316024

$RMP
[1] 6.204726e-20
```

4.5 Likelihood ratios

Likelihood ratios are computed using the **LR** function which implements the general formula of Curran *et al* for forensic DNA mixtures interpretation [13].

An example Consider the following genetic profiles from a rape case in Hong Kong [14]:

Locus	Mixture	Victim	Suspect	Frequency
D3S1358	14		14	0.033
	15	15		0.331
	17		17	0.239
	18	18		0.056

Table 1: Alleles from a DNA stain from a rape case in Hong Kong

Locus D3S1358 shows 4 distinct alleles (14, 15, 17 and 18), thus, the number of contributors to the mixed sample is taken 2.

Scenario 1 The following hypotheses are tested:

Prosecution hypotheses Hp: Contributors were the victim and the suspect.

Defense hypotheses Hd: Contributors were 2 unknown people.

First, the genotypes are assigned to the victim and the suspect:

```
> victim <- "15/18"  
> suspect <- "14/17"
```

Then, the likelihood ratio is computed using the **LR** function:

```
> LR(stain = c(14, 15, 17, 18), freq = c(0.033, 0.331, 0.239, 0.056),  
+     xp = 0, Tp = c(victim, suspect), Vp = NULL, Td = victim,  
+     Vd = NULL, xd = 2)
```

```
[1] 37.95501
```

The mixture profile is nearly 38 times more likely if it came from the suspect and the victim than if it came from two unknown unrelated individuals from the population of Hong Kong.

Scenario 2 The following hypotheses are tested:

Prosecution hypotheses Hp: Contributors were the victim and the suspect.

Defense hypotheses Hd: Contributors were the victim and one unknown.

```
> LR(stain = c(14, 15, 17, 18), freq = c(0.033, 0.331, 0.239, 0.056),  
+     xp = 0, Tp = c(victim, suspect), Vp = NULL, Td = victim,  
+     Vd = suspect, xd = 1)
```

```
[1] 63.39546
```

The mixture profile is 63 times more likely if it came from the suspect than if it came from an unrelated individual from the population of Hong Kong.

5 Two-person DNA mixtures resolution using allele peak heights or areas information: The *mastermix* interface

mastermix is a Tcl/Tk graphical user interface dedicated to the resolution of two-person DNA mixtures using allele peak heights or areas information. **mastermix** is the implementation of a method developed by Gill *et al* [15] and previously programmed into an Excel macro by Dr. Peter Gill.

This method searches through simulation the most likely combination(s) of the contributors' genotypes. Having previously obtained an estimation for the mixture proportion, it is possible to reduce the number of possible genotype combinations by keeping only those supported by the observed data. This is achieved by computing the sum of square differences between the expected allelic ratio and the observed allelic ratio, for all possible mixture combinations. The likelihood of peak heights

(or areas), given the combination of genotypes, is high if the residuals are low. Genotype combinations are thus selected according to the peak heights with the highest likelihoods. Appendix A gives the formulas for the expected allelic ratios following from [15].

Typing `mastermix()` in the R console launches a dialog window (Figure 1):



Figure 1: The mastermix interface

mastermix offers a graphical representation of the simulation for three models:

- The two allele model: at a given locus, two alleles are observed in the DNA stain
- The three allele model: at a given locus, three alleles are observed in the DNA stain
- The four allele model: at a given locus, four alleles are observed in the DNA stain

A left-click on each button launches a simulation dialog window for the corresponding model, while a right-click opens the corresponding help page. For instance, a left-click on the “Two-allele model” button yields Figure 2:

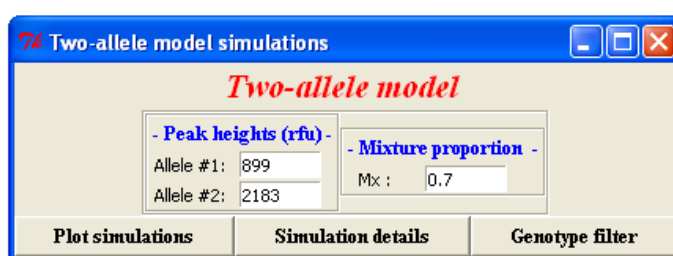


Figure 2: Two-allele model interface.

Note that default values for peak heights and observed mixture proportion are only given for illustration purposes.

As an example, we suppose that a locus showing four distinct alleles gives an estimation for the mixture proportion of 0.70, and that another locus shows two distinct alleles with heights of 899 and 2183 rfus. A left-click on the “Plot simulations” button yields a graphical representation of the residuals of each possible genotype combinations of the peak areas, for varying values of the mixture proportion across the interval $[0.1, 0.9]$.

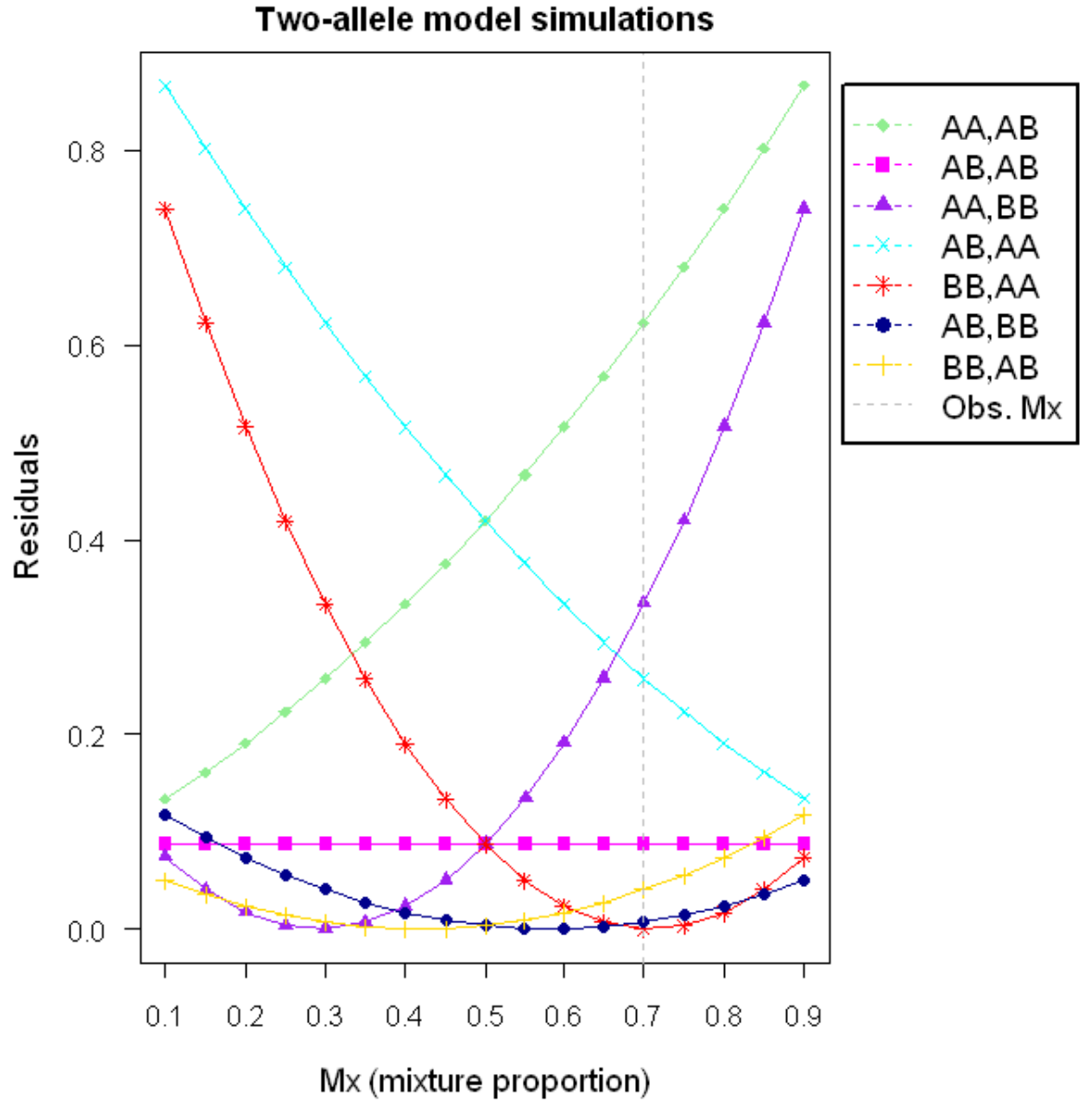


Figure 3: Graphical simulations of the residuals for each possible genotype combination, in a two-allele model, for every possible mixture combination based on variation of the mixture proportion.

The graphical simulation shows that multiple combinations correspond to the lowest residual value. The corresponding numerical results are obtained by clicking the “Simulations details” button:

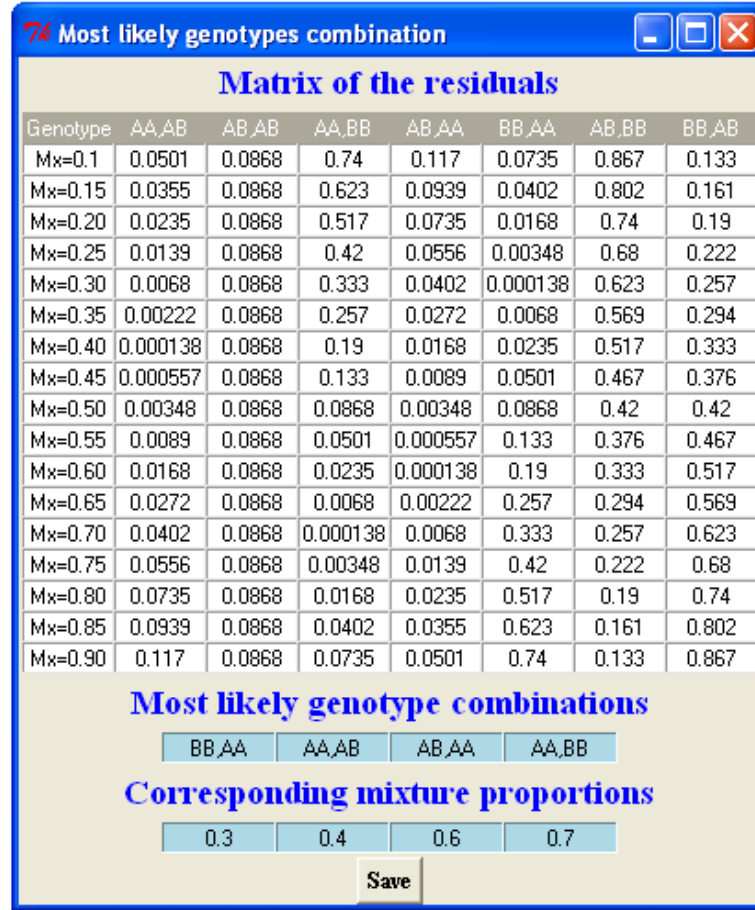


Figure 4: Numerical results of the graphical simulation.

Genotype combinations having the lowest residuals are highlighted along with the corresponding mixture proportion. The most likely combinations are: (BB,AA), (AA, AB), (AB, AA), (AA, BB) with the corresponding mixtures proportions :0.3, 0.4, 0.5 and 0.7. Note that clicking the “Save” button launches a window where the desired path for the save file can be specified, default creates a text file in the current folder.

The third button, “Genotypes filter” launches a window showing a matrix of the mixture proportion conditional on the genotype combination.

Genotype	{Mx conditioned}
AA,AB	0.42
AB,AB	NA
AA,BB	0.71
AB,AA	0.58
BB,AA	0.29
AB,BB	1.4
BB,AB	0.42

Figure 5: Genotypes filter: Mixture proportion conditional on the genotypes combination.

The mixture proportions conditional on the genotype combination gives a supplementary indication for the reduction of the number of possible combinations: Genotypes with non plausible mixture proportions ranges are not kept. The results confirm that genotypes which have not been already selected during the graphical simulation step, are not supported by the data. Formulas used for the calculations are given in Appendix A.

6 Miscellaneous

6.1 Manipulating forensim objects

forensim objects are mainly formed by lists and data frames. Modification of the slots of an object can easily be done using operators '\$' (lists) or '[' (data frame and matrix). For example, we wish to modify the frequencies of a given locus, say FGA, in the **tabfreq** object **tupop**:

```
> tupop$tab$Tu$FGA
```

```
      18      19      19.2      20      21      22      22.2      23      23.2      24      25
0.0392 0.0686 0.0033 0.0458 0.0980 0.1765 0.0033 0.1961 0.0098 0.2222 0.1013
      25.2      26      26.2      27
0.0065 0.0131 0.0065 0.0098
```

Frequencies of alleles 18 and 27 are modified from 0.0392 and 0.0098 to 0.01 and 0.03 respectively:

```
> tupop$tab$Tu$FGA[c("18", "27")] <- c(0.01, 0.03)
> tupop$tab$Tu$FGA
```

```
      18      19      19.2      20      21      22      22.2      23      23.2      24      25
0.0100 0.0686 0.0033 0.0458 0.0980 0.1765 0.0033 0.1961 0.0098 0.2222 0.1013
      25.2      26      26.2      27
0.0065 0.0131 0.0065 0.0300
```

6.2 How to change population names

Changing population names in any forensim object is achieved using the function `changepop`. For example, changing the population name in the `tabfreq` object `tupop` from “Tu” (argument `oldpop`) to “Tu2” (argument `newpop`) is achieved by:

```
> tupop2 <- changepop(tupop, oldpop = "Tu", newpop = "Tu2")
> tupop2@pop.names
```

```
[1] Tu2
Levels: Tu2
```

6.3 How to find the allele frequencies of a mixture

The allele frequencies of a mixture; stored in a `simumix` object, can be found using the function `findfreq`. The `tabfreq` object from which to extract the allele frequencies must be specified. For instance, allele frequencies in object `mix3` are found from the Caucasian population:

```
> temp <- findfreq(mix3, freq = strusa, reipop = "Cauc")
> temp
```

```
$Cauc
$Cauc$CSF1P0
      10      11      12
0.21689 0.30132 0.36093

$Cauc$FGA
      19      21      22      24      25
0.05298 0.18543 0.21854 0.13576 0.07119

$Cauc$TH01
      6      7      9      9.3
0.23179 0.19040 0.11424 0.36755

$Cauc$TPOX
      8      10      11
0.53477 0.05629 0.24338

$Cauc$VWA
      15      16      17      18
0.11093 0.20033 0.28146 0.20033

$Cauc$D3S1358
      15      16      17      18      19
0.26159 0.25331 0.21523 0.15232 0.01159

$Cauc$D5S818
      9      11      12
0.04967 0.36093 0.38411

$Cauc$D7S820
      8      9      12      13
0.15066 0.17715 0.16556 0.03477

$Cauc$D8S1179
      12      13      14      15
0.18543 0.30464 0.16556 0.11424

$Cauc$D13S317
      8      10      11      12
0.11258 0.05132 0.33940 0.24834
```

```

$Cauc$D16S539
      10      11      12
0.05629 0.32119 0.32616

$Cauc$D18S51
      12      13      16      18
0.12748 0.13245 0.13907 0.07616

$Cauc$D21S11
      29      29.2      30      32.2
0.19536 0.00331 0.27815 0.08444

$Cauc$D2S1338
      17      18      19      23      25      26
0.18212 0.07947 0.11424 0.11755 0.09272 0.02980

$Cauc$D19S433
      14      14.2      15      15.2
0.36921 0.01821 0.15232 0.03477

```

temp is a list of a single element "Cauc", which contains also a list:

```
> class(temp$Cauc)
```

```
[1] "list"
```

Allele frequencies of locus TPOX for example, are given by:

```
> temp$Cauc$TPOX
```

```

      8      10      11
0.53477 0.05629 0.24338

```

6.4 The number of alleles in a mixture

The number of alleles in a `simumix` object can be determined by the function `nball`. The overall loci number of alleles in the 2-person mixture `mix2` is:

```
> nball(mix2, byloc = FALSE)
```

```
[1] 11
```

and the numbers of alleles per locus can be obtained by setting the argument `byloc` to `TRUE`:

```
> nball(mix2, byloc = TRUE)
```

```

D8S1179      TH01      FGA
      3          4          4

```

References

- [1] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.
- [2] R Development Core Team. R : A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL [http : //www.Rproject.org/](http://www.Rproject.org/). 2006.
- [3] B. Zhu, J. Yan, C. Shen, T. Li, Y. Li, X. Yu, X. Xiong, H. Muf, Y. Huang, and Y. Deng. Population genetic analysis of 15 STR loci of Chinese Tu ethnic minority group. *Forensic Science International*, 174:255–258, 2008.
- [4] N. L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions, vol. 2*. John Wiley & Sons, 1995.
- [5] G. Nicholson, A. V. Smith, F. Jónsson, O. Gústafsson, K. Stefánsson, and P. Donnelly. Assessing population differentiation and isolation from single-nucleotide polymorphism data. *Journal of the Royal Statistical Society B*, 64:695–715, 2002.
- [6] J. Marchini and L. R. Cardon. Discussion on the meeting on "Statistical modelling and analysis of genetic data". *Journal of the Royal Statistical Society B*, 64:740–741, 2002.
- [7] D. R. Paoletti, T. E. Doom, C. M. Krane, M. L. Raymer, and D. E. Krane. Empirical analysis of the STR profiles resulting from conceptual mixtures . *Journal of Forensic Sciences*, 50(6):1361–1366, 2005.
- [8] J.M. Butler, R. Schoske, M.P. Vallone, J. W. Redman, and M. C. Kline. Allele frequencies for 15 autosomal STR loci on U.S. Caucasian, African American, and Hispanic populations. *Journal of Forensic Sciences*, 48(8):908–911, 2003.
- [9] J. M. Curran, C. M. Triggs, J. Buckleton, and B. S. Weir. Interpreting DNA Mixtures in Structured Populations. *Journal of Forensic Sciences*, 44(5):987–995, 1999.
- [10] H. Haned, D. Pontier, J. R. Lobry, L. Pene, and A. B. Dufour. Estimating the number of contributors to forensic DNA mixtures: does maximizing the likelihood performs better than the maximum allele count ? *Submitted*, 2009.
- [11] J. Buckleton, C. M. Triggs, and S. J. Walsh. *Forensic DNA evidence interpretation*. CRC PRESS, 2005.
- [12] D. J. Balding and R. A. Nichols. DNA profile match probability calculation: how to allow for population stratification, relatedness, databse selection and single bands. *Forensic Science International*, 64:125–140, 1994.
- [13] J. Curran, J. Buckleton, and C. M. Triggs. What is the magnitude of the subpopulation effect? *Forensic Science International*, 135:1–8, 2003.

- [14] W. K. Hu and W. K. Fung. Interpreting dna mixtures with the presence of relatives. *International Journal of Legal Medicine*, 117:39–45, 2003.
- [15] P. Gill, P. Sparkes, R. Pinchin, Clayton, J. Whitaker, and J. Buckleton. Interpreting simple STR mixtures using allele peak areas. *Forensic Science International*, 91:41–53, 1998.
- [16] T. Clayton and J. Buckleton. *Forensic DNA evidence interpretation*, chapter Mixtures, pages 217–239. CRS PRESS, 2005.

A Appendix: Formulas used in *mastermix*

A.1 Expected allelic ratios

Two-allele model: expected allelic ratios conditional on each possible genotype combination of the contributors to the mixture, when two alleles, A and B (in ascending order of molecular weights) are observed at a given locus, and \hat{M}_x is the proportion of sample from the first contributor [15].

Combination	Alleles	
	A	B
AA,AB	$\frac{\hat{M}_x}{2} + 0.5$	$\frac{1 - \hat{M}_x}{2}$
AB,AB	0.5	0.5
AA,BB	\hat{M}_x	$1 - \hat{M}_x$
AB,AA	$1 - \frac{\hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$
BB,AA	$1 - \hat{M}_x$	\hat{M}_x
AB,BB	$\frac{\hat{M}_x}{2}$	$1 - \frac{\hat{M}_x}{2}$
BB,AB	$\frac{1 - \hat{M}_x}{2}$	$\frac{\hat{M}_x}{2} + 0.5$

Three-allele model: expected allelic ratios conditional on each possible genotype combination of the contributors to the mixture when three alleles, A, B and C (in ascending order of molecular weights) are observed at a given locus [15].

Combination	Alleles		
	A	B	C
AA,BC	\hat{M}_x	$\frac{1 - \hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$
BB,AC	$\frac{1 - \hat{M}_x}{2}$	\hat{M}_x	$\frac{1 - \hat{M}_x}{2}$
CC,AB	$\frac{1 - \hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$	\hat{M}_x
AB,AC	0.5	$\frac{\hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$
BC,AC	$\frac{1 - \hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$	0.5
AB,BC	$\frac{\hat{M}_x}{2}$	0.5	$\frac{1 - \hat{M}_x}{2}$
BC,AA	$1 - \hat{M}_x$	$\frac{\hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$
AC,BB	$\frac{\hat{M}_x}{2}$	$1 - \hat{M}_x$	$\frac{\hat{M}_x}{2}$
AB,CC	$\frac{\hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$	$1 - \hat{M}_x$
AC,AB	0.5	$\frac{1 - \hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$
AC,BC	$\frac{\hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$	0.5
BC,AB	$\frac{1 - \hat{M}_x}{2}$	0.5	$\frac{\hat{M}_x}{2}$

Four-allele model: expected allelic ratios conditional on each possible genotype combination of the contributors to the mixture when four alleles, A, B, C and D (in ascending order of molecular weights) are observed at a given locus [15].

Combination	Alleles			
	A	B	C	D
AB,CD	$\frac{\hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$
AC,BD	$\frac{\hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$
AD,BC	$\frac{\hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$
BC,AD	$\frac{1 - \hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$
BD,AC	$\frac{1 - \hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$
CD,AB	$\frac{1 - \hat{M}_x}{2}$	$\frac{1 - \hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$	$\frac{\hat{M}_x}{2}$

A.2 Conditional mixtures proportions

The following tables give the formulas for the mixture proportion conditional on the genotype combinations. The conditional mixture proportions are computed using observed allele peak heights (or equivalently peak areas) [16].

Mixture proportions conditioned on the genotype combination for a locus showing two alleles, A and B (in ascending order of molecular weights), with peak heights ϕ_A and ϕ_B .

Two-allele model

Genotype combination	Conditional mixture proportion
AA,AB	$\frac{\phi_A - \phi_B}{\phi_A + \phi_B}$
AB,AB	No information is present
AA,BB	$\frac{\phi_A}{\phi_A + \phi_B}$
AB,AA	$\frac{2\phi_B}{\phi_A + \phi_B}$
BB,AA	$\frac{\phi_B}{\phi_A + \phi_B}$
AB,BB	$\frac{2\phi_A}{\phi_A + \phi_B}$
BB,AB	$\frac{\phi_B - \phi_A}{\phi_A + \phi_B}$

Mixture proportions conditioned on the genotype combination for a locus showing three alleles,, A , B and C (in ascending order of molecular weights), with peak heights ϕ_A , ϕ_B and ϕ_C .

Three-allele model

Genotype combination	Conditional mixture proportion
AA,BC	$\frac{\phi_A}{\phi_A + \phi_B + \phi_C}$
BB,AC	$\frac{\phi_B}{\phi_A + \phi_B + \phi_C}$
CC,AB	$\frac{\phi_C}{\phi_A + \phi_B + \phi_C}$
AB,AC	$\frac{\phi_B}{\phi_B + \phi_C}$
BC,AC	$\frac{\phi_B}{\phi_A + \phi_B}$
AB,BC	$\frac{\phi_A}{\phi_A + \phi_C}$
BC,AA	$\frac{\phi_B + \phi_C}{\phi_A + \phi_B + \phi_C}$
AC,BB	$\frac{\phi_A + \phi_C}{\phi_A + \phi_B + \phi_C}$
AB,CC	$\frac{\phi_A + \phi_B}{\phi_A + \phi_B + \phi_C}$
AC,AB	$\frac{\phi_C}{\phi_B + \phi_C}$
AC,BC	$\frac{\phi_A}{\phi_A + \phi_B}$
BC,AB	$\frac{\phi_C}{\phi_A + \phi_C}$

Mixture proportions conditioned on the genotype combination for a locus showing four alleles, A , B, C and D (in ascending order of molecular weights), with peak heights ϕ_A , ϕ_B , ϕ_C and ϕ_D .

Four-allele model

Genotype combination	Conditional mixture proportion
AB,CD	$\frac{\phi_A + \phi_B}{\phi_A + \phi_B + \phi_C + \phi_D}$
AC,BD	$\frac{\phi_A + \phi_C}{\phi_A + \phi_B + \phi_C + \phi_D}$
AD,BC	$\frac{\phi_A + \phi_D}{\phi_A + \phi_B + \phi_C + \phi_D}$
BC,AD	$\frac{\phi_B + \phi_C}{\phi_A + \phi_B + \phi_C + \phi_D}$
BD,AC	$\frac{\phi_B + \phi_D}{\phi_A + \phi_B + \phi_C + \phi_D}$
CD,AB	$\frac{\phi_C + \phi_D}{\phi_A + \phi_B + \phi_C + \phi_D}$