A tutorial for the R package *forensim*

Hinda Haned

May 5, 2009

Contents

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 and give practical and reproducible examples. Second, I present the statistical tools for forensic DNA mixtures interpretation. Third, various functionalities of forensim are exposed.

2 Getting started

2.1 forensim installation

Last stable version can be obtained by typing the line command: Than, the package must be loaded:

```
> library(forensim)

   ### forensim 1.1.0 is loaded ###
```

2.2 How to get help

- Please ask questions on the forensim help mailing list: forensim-help@lists.r-forge.r-project.org
- forensim manual can be found at :

3 Importing and 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, from either one or several populations
- 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 or simulated data. Importing pre-existing data into forensim objects is achieved using specific functions called constructors, that have same names than the object they are linked to. These construcrors can also be used for data simulation as it will be shown in the next section.

3.1 tabfreq objects

In forensim, allele frequencies are stored in tabfreq objects. Importing data into tabfreq objects is achieved using the tabfreq constructor. Input data must be a data frame or a matrix in the format of the *Journal of Forensic Sciences* for population genetic data: allele names are given in the first column, and frequencies for a given allele are read in rows for different loci. When a given allele is not observed, value is coded NA (instead of the original "-"):

```
> data(Tu)
> is.matrix(Tu)
```

```
[1] FALSE
```

```
> is.data.frame(Tu)
```

```
[1] TRUE
```

```
> head(Tu)
```

```
  Allele D8S1179 D21S11 D7S820 CSF1PO 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
```

Tu is a data frame giving allele frequencies for 15 short tandem repeats loci commonly used in forensic studies, in the Tu Chinese population (see ?Tu). This data frame is converted to a tabfreq object by the tabfreq constructor:

```
> tupop <- tabfreq(tab = Tu, pop.names = as.factor("Tu"))
> 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 the 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, which are optional only when a single population is handled, are stored in the @pop.names argument:

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

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

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

```
> tupop$which.loc
```

```
 [1] "D8S1179"  "D21S11"   "D7S820"   "CSF1PO"   "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 matrix) must be given as a list of data frames (or matrix) in the tab argument, and the pop.names argument becomes obligatory in order to distinguish the populations.

3.2 simugeno objects

simugeno objects are used to store pre-existing genotypes, or simulated genotypes from a tabfreq object. simugeno objects are created from tabfreq objects by specifying the desired number of individuals in argument n. By default, all loci in the tabfreq object are used, for the illustration purpose, 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:  individuals ID's
@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 the 10 simulated genotypes from the Tu population allele frequencies. For instance, genotypes of the five first simulated individuals at the two first loci are obtained by:

```
> tugeno$tab.geno[1:5, 1:2]
```

```
     D8S1179 TH01
ind1 "13/10" "8/7"
ind2 "11/14" "9/9"
ind3 "12/15" "9/7"
ind4 "13/13" "7/9.3"
ind5 "15/10" "7/9"
```

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) IDs:

```
> 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
Levels: Tu
```

3.3 simumix objects

simumix objects store DNA mixtures, only qualtitative data is handeled for the moment. simumix objects can be created from simugeno objects, given the number of contributors to the mixture:

```
> mix1 <- simumix(tugeno, ncontri = 2)
> mix1


   # Simumix object: simulated mixtures #


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

```
> mix1$mix.all


$D8S1179
[1] "10" "13" "15"

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

$FGA
[1] "22" "23" "24" "27"
```

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

```
> mix1@ncontri


[1] 2
```

and their genotypes.

```
> mix1$mix.prof


     D8S1179 TH01    FGA
ind5 "15/10" "7/9"   "22/24"
ind4 "13/13" "7/9.3" "23/27"
```

3.4 Allele frequencies simulation

We denote denote L a locus with k alleles and the ith allele frequency at this locus, in a given population, is denoted p_i.

3.4.1 The homogenous 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:

$$(p_1, ..., p_k) \rightsquigarrow Dirichlet(\alpha_1, ..., \alpha_k)$$

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 0.92465058 0.3214050 0.3395316 0.1486649 0.271448147
2      2 0.07534942 0.2175290 0.4048393 0.1340016 0.427651381
3      3         NA 0.4610661 0.1990252 0.4210536 0.115081583
4      4         NA        NA 0.0566039 0.0724661 0.071403658
5      5         NA        NA        NA 0.2238137 0.109265593
6      6         NA        NA        NA        NA 0.005149638
```

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.

3.4.2 The subdivided population case

Principle

The simupopD function simulate multi-population allele frequencies for independent loci, from a given reference population, following a Dirichlet model.
Allele frequencies in the populations are generated as random deviates from a Dirichlet distribution, which parameters control the deviation of allele frequencies from the average values in the reference population.
Allele frequencies in the subpopulations are generally not known, at least not with certainty, each allele frequency is modelled as a random variable; with a parameter α_i (ref):

$$\alpha_i = \frac{p_i(1-\theta)}{\theta}$$

where θ is Wright's *Fst* coefficient which allows accounting for population subdivision. The vector of allele frequencies at a given locus is obtained by:

$$(p_1, ..., p_k) \rightsquigarrow Dirichlet\left(\frac{p_1(1-\theta)}{\theta}, ..., \frac{p_k(1-\theta)}{\theta}\right)$$

An example

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

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

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

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 tools for forensic DNA mixtures interpretation

In forensim, two methods for the estimation of the number of contributors are implemented: the maximum allele count [3], and an estimator based on likelihood maximization [4].

4.1 The maximum allele count

This method consists in setting the lower bound of the number of contributors to a mixture to the minimum required to explain the observed profiles. 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, lets simulate a 3-person mixture from the strusa data set, using allele frequencies from the Caucasian population (see ?strusa):

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

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

```
> strusa$pop.names
```

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

The number of genotypes to simulate must be specified in each population in the 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. 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 function mincontri. This number can either be computed from all available loci simultaneously, in this (default) case the loc argument ise set to NULL:

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

```
[1] 3
```

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

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

```
[1] 3
```

4.2 The likelihood based 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* [5] for DNA mixtures interpretation, in the particular case where all contributors to the mixture are unknown and there are no typed individuals [4].

4.2.1 Likelihood of a mixture alleles conditional on the number of contributors

lik.loc computes the likelihood of observing a given mixture alleles knowing that there are x individuals contributing to the mixture, for a given locus. 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


   # Simumix object: simulated mixtures #


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

the likelihood per locus of observing the mixture 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
0.00000000 0.00000000 0.08397782 0.00000000 0.06244472 0.00000000 0.00000000
    D7S820    D8S1179    D13S317    D16S539     D18S51     D21S11    D2S1338
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
   D19S433
0.00000000
```

the likelihood that 3 individuals contributed is:

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


      CSF1PO          FGA         TH01         TPOX          VWA      D3S1358
2.127112e-01 1.439693e-02 1.003913e-02 1.176101e-05 3.151096e-03 4.084219e-03
      D5S818       D7S820       D8S1179      D13S317      D16S539       D18S51
2.576468e-02 7.138490e-03 9.696459e-03 7.933052e-02 4.886416e-03 3.971088e-04
      D21S11       D2S1338      D19S433
4.430904e-02 1.532551e-03 2.031781e-02
```

Note here that strusa contains three populations, so the reference population, here Caucasians, must be specified in the argument refpop.
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] 1.762545e-33
```

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] [4]. These functions give the number of contributors that maximizes the likelihood (max) and the corresponding likelihood value (maxvalue). The estimations per locus are:

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

          CSF1PO        FGA       TH01         TPOX        VWA     D3S1358
max    6.0000000 3.00000000 1.00000000 2.000000e+00 1.00000000 3.000000000
maxval 0.4875722 0.01439693 0.08397782 2.259076e-05 0.06244472 0.004084219
           D5S818     D7S820    D8S1179    D13S317    D16S539       D18S51
max    5.00000000 3.00000000 5.00000000 4.00000000 6.00000000 3.0000000000
maxval 0.03698673 0.00713849 0.02070741 0.09308793 0.01802073 0.0003971088
           D21S11     D2S1338   D19S433
max    2.00000000 2.000000000 2.0000000
maxval 0.06554738 0.001971789 0.0256489
```

and the estimation using all loci is:

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

               [,1]
max    3.000000e+00
maxval 1.762545e-33
```

4.3 The exclusion probability

The exclusion probability, also known as the Random Man Not Excluded (RMNE), is defined as "the probability that a random person would be excluded as a contributor to the mixture" [6], is implemented in forensim in the function PE.
PE 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 specified in the refpop argument. Implementation of PE includes the possibility of correcting for deviation from Hardy Weinberg proportions, due to population subdivision using Wright's *Fst*, called here theta [6]:

```
> PE(mix3, strusa, refpop = "Cauc", theta = 0, byloc = TRUE, digits = 2)

     CSF1PO FGA TH01 TPOX VWA D3S1358 D5S818 D7S820 D8S1179 D13S317 D16S539
PE_l   0.72 0.9 0.85 0.94 0.9    0.86   0.72    0.9    0.83     0.8    0.77
     D18S51 D21S11 D2S1338 D19S433
PE_l   0.94   0.86    0.94     0.8
```

argument byloc indicates if the exclusion probability should be computed per locus:

```
> PE(mix = mix3, freq = strusa, refpop = "Cauc", theta = 0, byloc = FALSE,
+     digits = 2)

PE
 1
```

Option digits correspond to the number of digits to show, default is set to 2 digits:

```
> PE(mix = mix3, freq = strusa, refpop = "Cauc", theta = 0, byloc = FALSE,
+     digits = 13)

PE
 1
```

5 Miscellaneous

5.1 Manipulating forensim objects

forensim objects are mainly formed by lists and data frames. Modification of an object slots can easily be done using operators '$' (lists) or '[' (data frame and matrix).

```
> tupop


   # Tabfreq object: allele frequencies  #


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

For example, we wish to modify the frequencies of a given locus, say FGA, for alleles 18 and 27 to 0.01 and 0.03 respectively:

```
> 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


> 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
```

Changing population names in any forensim object is achieved using function changepop:

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


[1] Tu2
Levels: Tu2
```

5.2 How to find the frequencies of a mixture alleles

Allele frequencies in a simumix object can be found from a tabfreq object using function findfreq. For instance, allele frequencies of locus TPOX in mix3 are found from the Caucasian population:

```
> findfreq(mix3, freq = strusa, refpop = "Cauc")$Cauc$TPOX


     10      11      12       5
0.05629 0.24338 0.04139 0.00166
```

5.3 The number of alleles in a mixture

The number of alleles in a simumix object can be determined by the nball function:

```
> nball(mix1, byloc = FALSE)
```

```
[1] 10
```

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

```
> nball(mix1, byloc = TRUE)
```

```
D8S1179    TH01     FGA
      3       3       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/. 2006.

[3] 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.

[4] 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 ? *In preparation*, 2009.

[5] 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.

[6] J. Buckleton, C. M. Triggs, and S. J. Walsh. *Forensic DNA evidence interpretation*. CRC PRESS, 2005.