

euroMix tutorial, version 1.1

Thore Egeland and Guro Dørum

July 8, 2013

This document gives an introduction to the R software package **euroMix** developed for analysing mixtures while accounting for related contributors and linkage. The package contains functions for simulating mixtures with relatives for both linked and unlinked markers, and for computing likelihood ratios conditioned on pedigrees. **euroMix** relies heavily on the R package **paramlink** (Egeland et al., 2013). Simulation of mixtures with relatives for linked markers uses MERLIN, while the R version of **Familias** is used to compute likelihood ratios that also accommodate theta-correction, mutation models and silent allele frequencies. Previous literature and methods have assumed the relationships between typed contributors to be same for the competing hypotheses. This restriction does not apply for our approach. The calculations are generalised to allow for general, possibly inbred, pedigrees. There is also a function **pvalue.machine** that calculates tail probabilities for LR_s. Planned extensions include handling of continuous data and artefacts, and to accommodate drop-in and drop-out in the computation of likelihood ratios.

euroMix is developed in close cooperation with the R package **forensim** (Haned, 2011) and this document is inspired by the similar documentation for **forensim**. It should however be noted that **forensim** is a much larger program and project compared to **euroMix**. **euroMix** is designed for users familiar with R and is not intended to be user friendly for other users.

euroMix version 1.1 is available from the R-Forge repository (<http://r-forge.r-project.org>), from which versions for all platforms (Windows, Mac OS X, Linux) can be downloaded. In the following sections we describe how to install **euroMix** and illustrate with examples how to use the main functions. Some very specialised functions considered to be of little relevance to users are not covered in this document, but the documentation is available online.

1 Installation

euroMix requires that the freely available R software is installed. See <http://www.cran.r-project.org/> on how to install R. **euroMix** is available from the R-Forge repository and can be installed from the command window in R with the command

```
> install.packages("euroMix", repos="http://R-Forge.R-project.org")
```

After installation the package is loaded with the command

```
> library(euroMix)
```

The R package `paramlink` is automatically installed together with `euroMix`. Some functions in `euroMix` also relies on the R version of `Familias` and on `MERLIN`, both must be installed manually. `Familias` is available on CRAN (<http://cran.r-project.org>) while `MERLIN` can be downloaded from <http://www.sph.umich.edu/csg/abecasis/merlin/index.html>. It is important to make sure that `MERLIN` is correctly pointed to in the `PATH` environment as explained in the documentation of the `merlin` function in `paramlink`.

2 Compute the likelihood for mixtures

We will show with an example how to compute the likelihood ratio for a set of hypotheses where the evidence is a DNA mixture. The prosecution and defense hypotheses are

- H_P : The victim and the suspect contributed to the mixture.
- H_D : The victim and the father of the suspect contributed to the mixture.

We start by defining the alleles, their frequencies and the observed alleles in the mixture.

```
> alleles <- 1:4 # Defines alleles labelled 1, 2, 3 and 4
> afreq <- c(0.044, 0.166, 0.11, 0.68) # Assigns allele frequencies
> R <- 1:3 # Assigns mixture evidence alleles as 1/2/3
```

Next, we need to create pedigrees that account for the relationship between the contributors to the mixture. The pedigrees are created with functions from `paramlink`. We will briefly present some of the functions in `paramlink` here, more detailed information can be found in the `paramlink` documentation (<http://cran.r-project.org/web/packages/paramlink/paramlink.pdf>). The female victim is unrelated to the other contributors and is therefore created as a pedigree with a single member using the function `singleton`. The relationship between the male suspect and his father is accounted for by creating a nuclear family with the function `nuclearPed`. It is important that the victim is assigned an index that does not coincide with the indices of the individuals in the nuclear family, in this case she is given the index 4.

```
> x <- singleton(4, sex=2) # Victim pedigree, singleton female called 4
> y <- nuclearPed(1) # Suspect pedigree
```

The pedigree functions return a `linkdat` object, which is essentially a list of information linked to the pedigree. See the `paramlink` documentation for more information about `linkdat` objects.

The victim is genotyped to 1/2 and the suspect is genotyped to 3/3, while the suspect's father is unknown. The known genotypes of the victim (4) and

suspect (3) are put into a list. Each element in the list is given as (i, g_i) where i denotes the index of the individual and g_i denotes its genotype, i.e. (4,1,2) means that individual 4 has genotype 1/2.

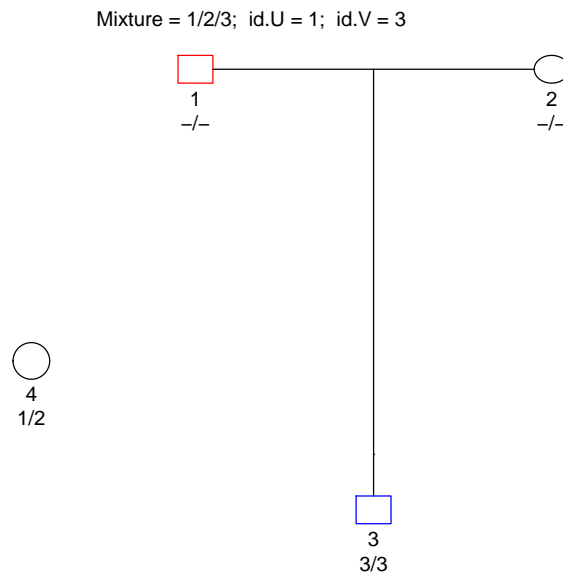
```
> known <- list(c(3,3,3), c(4,1,2)) #Known genotypes of 3 and 4
```

The likelihood of the mixture evidence is computed separately under the defense hypothesis and the prosecution hypothesis with the function `paraMix`. Under H_p there are no unknown contributors and this is signified by the argument `id.U=NULL`.

```
> lp <- paraMix(list(x,y), R, id.U=NULL, alleles=alleles, afreq=afreq,
+               known=known)$likelihood
```

Under H_d the father (1) is specified as an unknown contributor by setting `id.U=1`, while the suspect is specified as a known (genotyped) non-contributor by setting `id.V=3`. Note that we can choose to plot the pedigree (`plot=TRUE`) with the hypothesised contributors (in this case the suspect and father) marked.

```
> ld <- paraMix(list(x,y), R, id.U=1, id.V=3, alleles=alleles,
+               afreq=afreq, known=known, plot=TRUE)$likelihood
```



Finally, the likelihood ratio is computed as the ratio between the two likelihoods.

```
> lp/ld # Calculates required LR as 3.125
```

[1] 3.125

For likelihood ratio computations with theta correction, mutations and silent alleles, the function `famMix`, based on `Familias`, can be used. `Familias` first needs to be loaded into R. We will consider the same example as above, but with theta correction. Unlike `paraMix`, `famMix` can not handle singletons, i.e. pedigrees with only one individual, and unrelated individuals must therefore be specified as part of the pedigree.

```
> library(Familias)
> theta <- 0.01
> x <- nuclearPed(1)
> x <- addOffspring(x, father=1, noff=1)
```

In `famMix`, the known genotypes of 3 and 4 are specified as a marker object created with the function `marker` in `paramlink`. The likelihood is computed separately under H_p and H_d . The marker object with the known genotypes is specified in `famMix` through the argument `partialmarker`, while `id.U` and `id.V` are specified as for `paraMix`.

```
> m1 <- marker(x, 4, c(1,2), 3, c(3,3), alleles=alleles, afreq=afreq)
> lp <- famMix(x, R, id.U=NULL, partialmarker=m1,
+ theta=theta)$likelihood
> ld <- famMix(x, R, id.U=1, id.V=3, partialmarker=m1,
+ theta=theta)$likelihood
> lp/ld
```

[1] 2.886771

3 Compute a p-value corresponding to LR

The function `pvalue.machine` computes a p-value corresponding to a given likelihood ratio. The p-value is the probability of observing a likelihood ratio under H_d that is at least as large as the one observed. There is also a version of `pvalue.machine`, `pvalue.machineC`, that uses C code that may speed up the computations considerably.

We will consider an example with nine markers where we allow for drop-in and drop-out in the evidence. The hypotheses to be considered are

- H_P : The victim and the suspect contributed to the mixture.
- H_D : The victim and an unrelated unknown contributed to the mixture.

We start by loading data integrated in `euroMix`, including an allele frequency database (`db2`), a mixture and genotype profiles for a victim and a suspect (all found in the data object `'ex9m'`).

```

> data(db2)
> data(ex9m)
> M <- nrow(R)
> markers <- rownames(R)

```

The likelihood ratio is computed with the LR function in `forensim`. LR can handle drop-in and drop-out but not related contributors, but in this example there are no related contributors. It is required that `forensim` has been installed in advance (see <http://forensim.r-forge.r-project.org>) and that we load it into R. We use a drop-in probability of 0.47 and a drop-out probability of 0.05. A loop goes through all markers and computes the LR for each marker. Finally, the total LR is found as the product of the LR for each marker.

```

> library(forensim)
> lr <- A <- numeric()
> for(i in 1:M){
+   r <- as.numeric(R[i,!is.na(R[i,])])
+   v <- as.numeric(V[i,])
+   s <- as.numeric(S[i,])
+   afreq <- db2[db2[,1]==markers[i],3]
+   A[i] <- choose(length(afreq)+1,2)
+   names(afreq) <- db2[db2[,1]==markers[i],2]
+   lr[i] <- LR(r, Tp=c(v,s), Td=v, Vp=NULL, Vd=s,
+               xp=0, xd=1, theta=0, prDHet=rep(0.47,4),
+               prDHom=rep(0.47,4)^2, prC=0.05, freq=afreq)$LR
+ }
> LR.suspect <- prod(lr)

```

Next, we need to compute the LRs for all genotypes that may occur under H_d , that is, the LRs we get when we replace the suspect's profile with all genotype profiles that an unrelated unknown may have. At the same time we find the probability for each genotype of occurring. One loop goes through the markers and a second loop goes through the genotypes for each marker. The results are a matrix of LRs where the likelihood ratios are sorted in descending order within each row, and a matrix of corresponding probabilities.

```

> LR.table <- P.table <- matrix(0,M,max(A))
> for(i in 1:M){
+   afreq <- db2[db2$Marker==markers[i],]$Frequency
+   names(afreq) <- db2[db2$Marker==markers[i],]$Allele
+   #All possible genotypes for the given marker
+   gtAll <- as.matrix(expand.grid(names(afreq),names(afreq)))
+   gtAll <- gtAll[gtAll[,1]<=gtAll[,2],]
+   #Compute Likelihood ratios and corresponding
+   #genotype probabilities for each genotype
+   lrAll <- pAll <- numeric()
+   for(j in 1:nrow(gtAll)) {

```

```

+   rm <- gtAll[j,]
+   r <- as.numeric(R[i,!is.na(R[i,])])
+   v <- as.numeric(V[i,])
+   s <- as.numeric(S[i,])
+   lrAll[j] <- LR(r, Tp=c(v,rm), Td=v, Vp=NULL, Vd=rm, xp=0,
+                 xd=1, theta=0, prDHet=rep(0.47,4),
+                 prDHom=rep(0.47,4)^2, prC=0.05, freq=afreq)$LR
+   pAll[j] <- ifelse(rm[1]==rm[2], prod(afreq[as.character(rm)]),
+                     2*prod(afreq[as.character(rm)]) )
+ }
+ #Sort LR's in decreasing order
+ idxSort <- order(lrAll, decreasing=TRUE)
+ LR.table[i,1:length(lrAll)] <- lrAll[idxSort]
+ P.table[i,1:length(lrAll)] <- pAll[idxSort]
+ }

```

Finally, the p-value is computed with `pvalue.machine` by inserting the observed LR, the matrix of LR's and the matrix of probabilities.

```
> pvalue.machine(LR.suspect, LR.table, P.table)
```

```
[1] 1.399238e-09
```

4 Simulate mixtures and generate all possible genotypes

The function `simMixParamlink` can be used to simulate DNA mixtures. Given a `linkdat` object that contains marker data for the individuals of interest, it generates a mixture of the genotypes. We create a `linkdat` object with one of the pedigree functions and an empty marker object. Next, genotypes for individual 1 and 3 are simulated with the `markerSim` function in `paramlink` and added to the `linkdat` object. A mixture of the genotypes of individual 1 and 3 is generated by running `simMixParamlink` on the `linkdat` object. In this case the mixture consists of only the allele 4.

```

> alleles <- 1:4 # Defines alleles labelled 1, 2, 3 and 4
> afreq <- c(0.044, 0.166, 0.11, 0.68) # Assigns allele frequencies
> x <- nuclearPed(2) #Create pedigree
> m1 <- marker(x,alleles=alleles,afreq=afreq) #Create empty marker
> y <- markerSim(x,N=1,available=c(1,3),partialmarker=m1,
+               verbose=FALSE,loop=7,seed=2) #Simulate genotypes
+                                           #for individuals 1 and 3
> simMixParamlink(y,alleles)

```

```
[[1]]
```

```
[1] 4
```

The function **generate** creates all genotypes that an unknown contributor may have. It takes as input a mixture, the alleles of the known contributors and the number of unknown contributors. The result is a matrix where each row represent an unknown contributor and columns represent the alleles, columns one and two is the first genotype, columns three and four the second genotype, and so on.

```
> R <- 1:3 #Alleles in the mixture
> K <- 1:2 #Alleles of known contributors
> n <- 1 #Number of unknown contributors
> generate(R=R,K=K,x=n)
```

```
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]     3     3     1     3     2     3
```

References

- [1] H. Haned. Forensim: An open-source initiative for the evaluation of statistical methods in forensic genetics. *Forensic Science International: Genetics*, 5(4):265 – 268, 2011.
- [2] T. Egeland, N. Pinto and M.D. Vigeland. A general approach to power calculation for relationship testing. *Forensic Science International: Genetics*, doi: <http://dx.doi.org/10.1016/j.fsigen.2013.05.001> 2013.