Deconvolution of MPS STR DNA mixtures and approximation of the likelihood ratio

Søren B. Vilsen

2020-12-21

1 Introduction

The central question in forensic genetics is determination of the posterior odds of two competing hypotheses. Assume DNA evidence, $\mathcal{E}$, has been collected at a crime-scene. Furthermore, if we let $\mathcal{H}_p$ and $\mathcal{H}_d$ denote the hypothesis of the prosecutions and defences, respectively, then the posterior odds are given by:

$$\frac{\mathbb{P}\left(\mathcal{H}_d|\mathcal{E}\right)}{\mathbb{P}\left(\mathcal{H}_p|\mathcal{E}\right)} = \frac{\mathbb{P}\left(\mathcal{E}|\mathcal{H}_d\right)}{\mathbb{P}\left(\mathcal{E}|\mathcal{H}_p\right)} \frac{\mathbb{P}\left(\mathcal{H}_d\right)}{\mathbb{P}\left(\mathcal{H}_p\right)}. \tag{1}$$

The prior probabilities of the two hypotheses are either assumed equal or supplied by the court, which leaves us with the likelihood ratio:

$$\mathrm{LR}(\mathcal{H}_d, \mathcal{H}_p) = \frac{\mathbb{P}\left(\mathcal{E}|\mathcal{H}_d\right)}{\mathbb{P}\left(\mathcal{E}|\mathcal{H}_p\right)}. \tag{2}$$

Assuming that the evidence $\mathcal{E}$ consists of some quantitative information $\boldsymbol{\mathcal{Q}}$ (the coverage in MPS or peak height/area in CE) about some genetic information $\boldsymbol{g}$, we can factorise $\mathbb{P}\left(\mathcal{E}|\mathcal{H}_i\right)$ as:

$$\mathbb{P}\left(\mathcal{E}|\mathcal{H}_i\right) = \mathbb{P}\left(\boldsymbol{\mathcal{Q}}, \boldsymbol{g}|\mathcal{H}_i\right) = \mathbb{P}\left(\boldsymbol{\mathcal{Q}}|\boldsymbol{g}\right) \mathbb{P}\left(\boldsymbol{g}|\mathcal{H}_i\right). \tag{3}$$

Thus, the probability of the evidence, given a hypothesis, has two parts: (1) The probability of the quantitative information given allelic information, and (2) The probability of the allelic information under the hypothesis in question. The population is assumed either in Hardy-Weinberg equilibrium (HWE) or it is assumed to be inbred, which is accounted for using θ correction (Balding and Nichols 1994).

If a hypothesis states that a contributor is unknown, it becomes necessary to sum over the set of possible unknown contributors to the mixture, in order to correctly state the probability of the evidence. That is, the probability of the evidence in the case of unknown contributors is:

$$\mathbb{P}\left(\mathcal{E}|\mathcal{H}_i\right) = \sum_{\boldsymbol{g}_u \in \mathcal{U}_i} \mathbb{P}\left(\boldsymbol{\mathcal{Q}}|\boldsymbol{g}_{K_i}, \boldsymbol{g}_u\right) \mathbb{P}\left(\boldsymbol{g}_u|\boldsymbol{g}_K\right),$$

where $\boldsymbol{g}_{K_i}$ are the known contributors under hypothesis i, $\boldsymbol{g}_K = \{\boldsymbol{g}_{K_d}, \boldsymbol{g}_{K_p}\}$, and $\mathcal{U}_i$ is the set of possible unknown contributors under hypothesis i.

The sum over the set of unknown contributors may be intractable. Therefore, we search through the space of unknown contributors using an Evolutionary Algorithm. The algorithm will have two objectives:

(1) Find the combination of unknown genotypes maximising the $\mathbb{P}\left(\boldsymbol{\mathcal{Q}}|\boldsymbol{g}_{K_i}, \boldsymbol{g}_u\right) \mathbb{P}\left(\boldsymbol{g}_u|\boldsymbol{g}_K\right)$, under an hypothesis $\mathcal{H}_i$.

(2) Approximate the likelihood ratio by approximating the set of combinations of unknown contributors by the elements having the largest contribution to the sum.

2 The coverage model

We define the coverage model in two parts: (1) the allele coverage model, which will account for the alleles of the contributors and any systematic errors produced by the sequencing process, and (2) a noise coverage model accounting for the remaining observations.

The allele coverage model, introduced in (Vilsen et al. 2017), was adapted from capillary electrophoresis peak height/width models (Tvedebrink et al. 2013), (Taylor, Bright, and Buckleton 2013), (Cowell et al. 2015), (Bleka, Storvik, and Gill 2016), and (Steele, Greenhalgh, and Balding 2016), to model the allele coverage generated by the MPS process. The coverage of a string is the number of times the string is observed (i.e. the count). Therefore, we model the coverage as overdispersed count data. That is, the coverage of allele a marker m is distributed as:

$$y_{ma} \sim \text{PG1}\left(\mu_{ma}, \gamma\right),$$

where PG1 is the Poisson-gamma distribution (the mean parametrised negative binomial distribution) of order 1 (the variance is $\mu_{ma}(1+\gamma)$), μ_{ma} the expected coverage, and γ the overdispersion. Furthermore, we assume that the expected coverage is given as:

$$\mu_{ma} = \nu\beta_m \sum_c \left[g_{mac} + \sum_{A \in \mathcal{P}(a)} \xi_{mA} g_{mAc} \right] \varphi_c,$$

where ν is the average heterozygote coverage, β_m is a marker dependent scaling parameter, with $\sum \beta_m = M$, φ_c is the relative contribution of contributor c, with $\sum_c \varphi_c = 1$, g_{mac} is the genotype of marker m allele a contributor c, $\mathcal{P}(a)$ is the set of potential parents of allele a and ξ_{mA} is the stutter ratio of parent A. Note that the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\xi}$, as well as the set of potential parents are assumed known throughout this vignette.

Any observation not identified as an allele or a stutter of an allele is classified as noise and denoted using set complement notation: $\boldsymbol{y}^c$ and $\boldsymbol{g}^c$. The coverage of the noise is assumed to follow a zero truncated Poisson-gamma distribution with mean ψ and overdispersion ρ.

Assuming that the two parts are independent given the genotypic information, then the likelihood can be written as:

$$\mathbb{P}\left(\boldsymbol{\mathcal{Q}} | \boldsymbol{g}_{K_i}, \boldsymbol{g}_u\right) = \prod_m L(\nu, \eta, \boldsymbol{\varphi} | \beta_m, \boldsymbol{\xi}_m, \boldsymbol{y}_m, \boldsymbol{g}_m) L(\psi, \rho | \boldsymbol{y}_m^c, \boldsymbol{g}_m^c) \tag{4}$$

3 Example

We will need the following `R` packages:

```
library("Biostrings")
library("tidyverse")
library("stringr")
library("Rsolnp")
library("MPSMixtures")
library("microbenchmark")
```

First, we need to make some assumption about the population from which these alleles are drawn. That is, we need an allelic ladder of this population and we need to make assumptions on the allele frequencies and the inbreding coefficient. The `MPSMixtures` package has an example ladder and corresponding frequencies called `examplePopulation`. For simplicity, we will assume that any allele combination is equally likely. That is, eliminating the genotypic distribution entirely. This is done to make the deconvolution results easily comparable with the truth. In the package, the genotypic distribution is ignored, by assigning a negative value to the inbreeding coefficient (θ):

```
theta <- -1.0
data(examplePopulation)
```

An allelic ladder can also be generated using the `generateAllelicLadder` function, specifying the possible regions and their frequncies, using the arguments `markers`, `regions`, `frequencies`, and if necessary `motifLength`. Note: all three vectors need to be of equal length (or length 1). The code used to generate the `examplePopulation` can be seen in the example of the `generateAllelicLadder` function.

As we cannot use a real DNA profile as an example, we will simulate data given a fictional case of murder. To simulate the coverage, in this case of murder, we will assume the following: the marker imbalances, a stutter ratio model, the number of observed alleles for each marker, the true profiles of the contributors (which we will simulate), and the parameters of the model.

The number of markers is $M = 10$ and we sample the marker imbalances, $\boldsymbol{\beta}$, using a gamma distribution:

```
set.seed(123456)
markers <- unique(examplePopulation$Marker)
numberOfMarkers <- length(markers)

markerImbalances <- rgamma(numberOfMarkers, shape = 10, rate = 10)
markerImbalances <- markerImbalances / mean(markerImbalances)
markerImbalances
```

```
##  [1] 1.0372583 0.7343623 0.5987315 1.5003802 1.0374738 1.1840109 1.5909785
##  [8] 1.1387861 0.6973960 0.4806224
```

We will assume that stutter ratio is consistent across markers (which is very unrealistic), and create the stutter ratio model, as:

```
stutterRatioTibble <-
  tibble(BlockLengthMissingMotif = sample(1:40, 1000, replace = T),
         StutterRatio = 0.005 * BlockLengthMissingMotif + rnorm(1000, 0, 0.02)) %>%
  mutate(StutterRatio = ifelse(StutterRatio < 0, 0, StutterRatio))

stutterRatioModel <- lm(StutterRatio ~ 0 + I(BlockLengthMissingMotif - 1),
                        data = stutterRatioTibble)
```

Given the number of contributors, and the population data generated above, we sample the genotypes from the population assuming HWE.

```
numberOfContributors <- 2
trueProfiles <- sampleGenotypesHWE(numberOfContributors, examplePopulation, c("V", "P"))
```

The parameters in the allele coverage model are:

```
nu <- 1400
eta <- 8
phi <- c(0.7, 0.3)
```

The parameters in the noise coverage model are:

```
psi <- 3
rho <- 2
pi <- 0.5

noiseParameters <- c(psi, rho, pi)
```

The coverage can now be simulated using the `sampleCoverage` function, returning a `tibble` containing the sampled coverage:

```
sampleTibble <- sampleCoverage(
  trueProfiles = trueProfiles, markerImbalances = markerImbalances,
  populationLadder = examplePopulation, stutterRatioModel = stutterRatioModel,
  alleleCoverageParameters = list(nu = nu, eta = eta, phi = phi),
  noiseParameters = list(psi = psi, rho = rho, pi = pi, maxElements = 20),
  p = NULL)
```

Any user defined data-set must have the same structure and column names as the `sampleTibble` `tibble`. The `sampleTibble`-tibble is combined with `examplePopulation` in such a way that the regions in `examplePopulation` not seen in `sampleTibble` are added with a coverage of 0. This allows us to account for drop-outs in the model, while drawing on the allele frequencies for information. Note that any region attributed to the noise distribution with a coverage of zero is ignored.

We have written this package with two objectives in mind:

(1) Find the combination of unknown genotypes maximising the $L\left(\boldsymbol{y}|\boldsymbol{g}_{K_i}, \boldsymbol{g}_u\right)\mathbb{P}\left(\boldsymbol{g}_u|\boldsymbol{g}_K\right)$, under some hypothesis $\mathcal{H}_i$.

(2) Approximate the likelihood ratio by approximating the set of combinations of unknown contributors, by using the elements having the largest contribution to the sum.

Furthermore, we will build and store the list of potential parents. As the data does not change, we do not want to rebuild it every single time we initialise an algorithm:

```
potentialParentsList <- potentialParentsMultiCore(sampleTibble, stutterRatioModel)
```

Note: we can also calculate the fitness and estimate the parameters, assuming that both profiles are known, by calling the `estimateParametersOfKnownProfiles` function.

```
knownProfilesIndividual <-
  estimateParametersOfKnownProfiles(
    sampleTibble = sampleTibble, markerImbalances = markerImbalances,
    knownProfilesList = trueProfiles, potentialParentsList = potentialParentsList,
    noiseParameters = noiseParameters, stutterRatioModel = stutterRatioModel,
    levelsOfStutterRecursion = 1, convexMarkerImbalanceInterpolation = 0.8,
    tolerance = rep(1e-8, 4), numberOfThreads = 4
  )

knownProfilesIndividual[c("Parameters", "LogLikelihoods", "Fitness")]
```

```
## $Parameters
## $Parameters$SampleParameters
## [1] 1412.780541    6.221316
##
## $Parameters$MixtureParameters
## [1] 0.7053375 0.2946625
##
## $Parameters$MarkerImbalanceParameters
##  [1] 1.0483885 0.7297301 0.5656538 1.4902449 0.9930164 1.2714498 1.5369255
##  [8] 1.1619796 0.7210470 0.4815644
##
## $Parameters$NoiseParameters
## [1] 3.0 2.0 0.5
##
##
## $LogLikelihoods
## [1] -254.75660  -49.33696    0.00000
```

```
##
## $Fitness
## [1] -304.0936
```

The `MPSMixtures` offers two simple fit evaluation plots: Q-Q plots of the deviance residuals and prediction intervals for the expected coverage. They are created by calling the `ggplotQQPlotProfiles` and `ggplotPredictionIntervals` functions, respectively. The resulting figures can be seen in Figure 1 and 2.

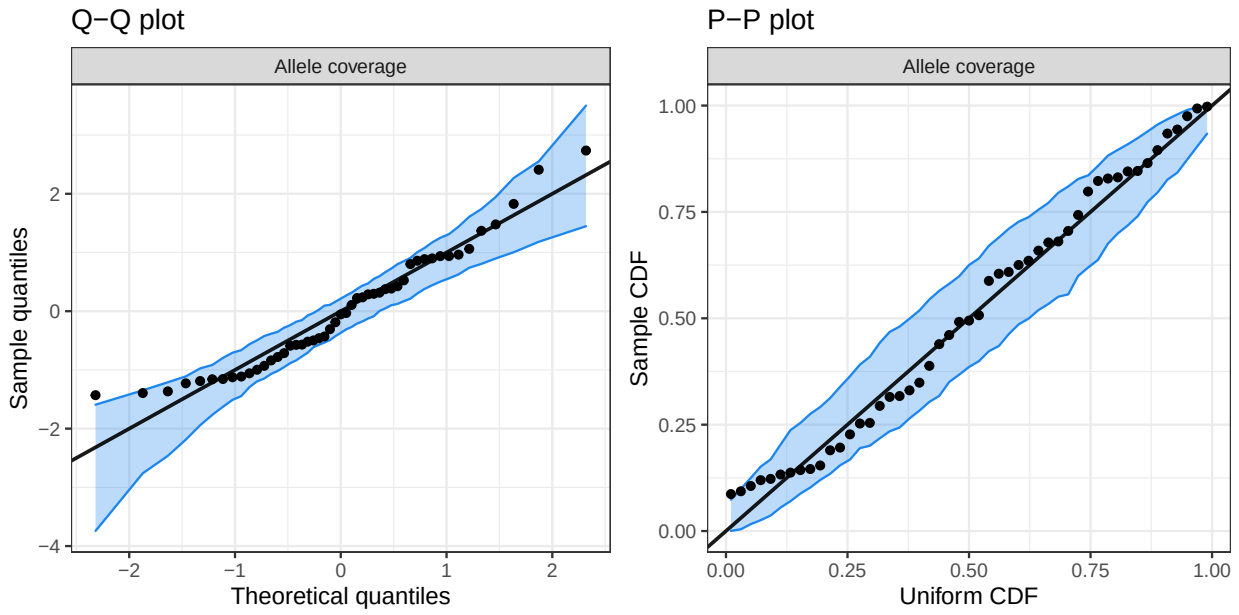

Figure 1: Q-Q plot of the fitted model for both the allele and noise coverage models.

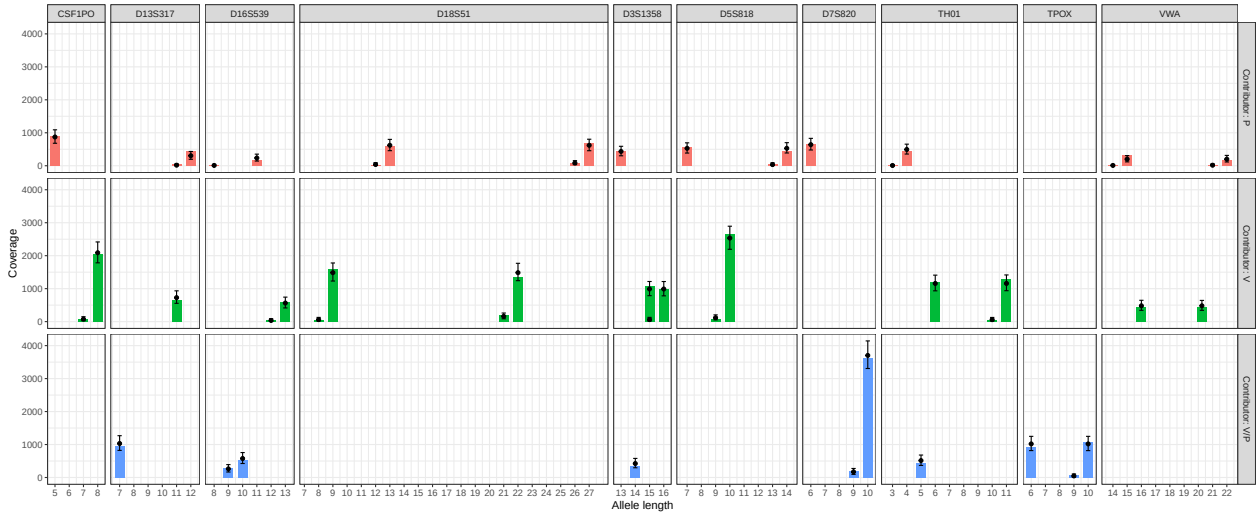

Figure 2: Bar-plot of the coverage against the allele length for the each marker and profile, shown in coloumns and rows, respectively. Furthermore, the prediction interval for each observation is seen as the black bars, with the expected coverage at their center.

3.1 Finding the best combination of unknown genotypes

As two is the total number of contributors, we have three possible spaces of unknown genotypes. The spaces, when (1) the minor (perpetrator) is known, (2) the major (victim) is known, and (3) both contributors are unknown.

We will take each case in turn, trying to ascertain the optimal genotype (or combination of genotypes), under the model. Furthermore, we will, only in the first case, use both a single population and multiple population model. In cases two and three, we will stick to the multiple population approach, as it is more than ten times faster.

Note that if the genotypic information (allele frequencies) in the population should be ignored, we can set $\theta < 0$ or set all allele frequencies to zero.

3.1.1 Known minor

We create an hypothesis using the `setHypothesis` function. The known profiles should passed as a list containing an element for each of the known profiles (in this case the perpetrator).

```
knownPerpetrator <- setHypothesis(sampleTibble, numberOfContributors, trueProfiles["P"], theta)
```

We start with the single population algorithm:

```
controlSinglePopulation <-
  optimalUnknownProfileCombination.control(
    numberOfPopulations = 1,
    numberOfIterations = 350,
    populationSize = 100,
    numberOfFittestIndividuals = 1,
    mutationDecayRate = 1,
    hillClimbingIterations = 0,
    parentSelectionWindowSize = 15,
    allowParentSurvival = TRUE,
    trace = FALSE,
    levelsOfStutterRecursion = 1,
    numberOfIterationsEqualMinMax = 10,
    tolerance = rep(1e-8, 4)
  )

singlePopulationBenchmarkMajor <- microbenchmark(
  optimalSingleMajor <-
    optimalUnknownProfileCombination(sampleTibble = sampleTibble,
                                     markerImbalances = markerImbalances,
                                     H = knownPerpetrator[[1]],
                                     potentialParentsList = potentialParentsList,
                                     noiseParameters = noiseParameters,
                                     control = controlSinglePopulation),
  times = 1)
```

The single population algorithm terminated in 18.69 seconds.

The fittest found individual is always listed as the first entry in the returned list. In this case, the log-likelihood in the fittest found individual is -304.0936, with the log-likelihoods of the allele, noise, and genotype probabilities at -254.7566, -49.337, and 0. That is, the found major contributor is the same as the true major contributor and the increase in fitness is solely due to the added uncertainty represented by the genotype probability. The corresponding estimated parameters are (the same as seen above):

```
optimalSingleMajor$U[[1]][["Parameters"]]
```

```
## $SampleParameters
## [1] 1412.7807    6.2213
##
## $MixtureParameters
## [1] 0.2946626 0.7053374
##
## $MarkerImbalanceParameters
##  [1] 1.0483885 0.7297301 0.5656538 1.4902449 0.9930164 1.2714498 1.5369255
##  [8] 1.1619796 0.7210470 0.4815644
##
## $NoiseParameters
## [1] 3.0 2.0 0.5
```

The relative difference for the parameters ν, η, ψ, ρ, and $\boldsymbol{\varphi}$ are given as: 0.01, 0.22, 0, 0, and 0.02, respectively. We see that the $\hat{\nu}$, $\hat{\gamma}$, $\hat{\psi}$, and $\hat{\boldsymbol{\varphi}}$ are all within 10% of the true values, only the overdispersion parameter of the noise distribution, $\hat{\rho}$, has proven difficult to estimate.

We will use 32 sub-populations spread on four cores. Furthermore, we set the size of each population to 75, i.e. a total population size of 2,400.

```
controlMultiplePopulation <-
  optimalUnknownProfileCombination.control(
    numberOfPopulations = 16,
    numberOfIterations = 100,
    populationSize = 50,
    numberOfFittestIndividuals = 1,
    numberOfIterationsEqualMinMax = 10,
    mutationDecayRate = 1,
    hillClimbingIterations = 0,
    parentSelectionWindowSize = 6,
    allowParentSurvival = TRUE,
    trace = FALSE,
    levelsOfStutterRecursion = 1,
    tolerance = rep(1e-8, 4)
  )

multiplePopulationBenchmarkMajor <- microbenchmark(
  optimalMultipleMajor <-
    optimalUnknownProfileCombination(sampleTibble = sampleTibble,
                                     markerImbalances = markerImbalances,
                                     H = knownPerpetrator[[1]],
                                     potentialParentsList = potentialParentsList,
                                     noiseParameters = noiseParameters,
                                     control = controlMultiplePopulation),
  times = 1)
```

The parallel EA terminated in 179.23 seconds, i.e. close to ten times faster than the single population algorithm. Thus, we see both an increased performance from the parallelisation and the reduction in population size. Furthermore, the fittest found individual had a fitness of -304.0936. That is, the two approaches have terminated with the same individual as the fittest individual.

In both cases the log-likelihoods and fitness of the fittest individual are given as:

```
optimalMultipleMajor$U[[1]][c("LogLikelihoods", "Fitness", "Parameters")]
```

```
## $LogLikelihoods
## [1] -254.75660  -49.33696    0.00000
##
## $Fitness
## [1] -304.0936
##
## $Parameters
## $Parameters$SampleParameters
## [1] 1412.7807    6.2213
##
## $Parameters$MixtureParameters
## [1] 0.2946626 0.7053374
##
## $Parameters$MarkerImbalanceParameters
##  [1] 1.0483885 0.7297301 0.5656538 1.4902449 0.9930164 1.2714498 1.5369255
##  [8] 1.1619796 0.7210470 0.4815644
##
## $Parameters$NoiseParameters
## [1] 3.0 2.0 0.5
```

As the parallel EA was both faster and got the exact same result, we will, in the remainder of this manuscript, only use the parallel EA.

3.1.2 Known major

Now let us assume that the major is known (i.e. the victim):

```
knownVictim <- setHypothesis(sampleTibble, numberOfContributors, trueProfiles["V"], theta)
```

We will use the same control settings, `controlMultiplePopulation`, as in the previous section.

```
optimalMultipleMinor <- optimalUnknownProfileCombination(sampleTibble = sampleTibble,
                                                         markerImbalances = markerImbalances,
                                                         H = knownVictim[[1]],
                                                         potentialParentsList = potentialParentsList,
                                                         noiseParameters = noiseParameters,
                                                         control = controlMultiplePopulation)
```

The parameters, log-likelihoods, and fitness are:

```
optimalMultipleMinor$U[[1]][c("Parameters", "LogLikelihoods", "Fitness")]
```

```
## $Parameters
## $Parameters$SampleParameters
## [1] 1412.780541    6.221316
##
## $Parameters$MixtureParameters
## [1] 0.7053375 0.2946625
##
## $Parameters$MarkerImbalanceParameters
##  [1] 1.0483885 0.7297301 0.5656538 1.4902449 0.9930164 1.2714498 1.5369255
##  [8] 1.1619796 0.7210470 0.4815644
##
## $Parameters$NoiseParameters
## [1] 3.0 2.0 0.5
```

```
##
##
## $LogLikelihoods
## [1] -254.75660  -49.33696    0.00000
##
## $Fitness
## [1] -304.0936
```

Comparing the log-likelihoods of the optimal minor contributor to those of the true profile, we see that the log-likelihood of the noise model does not change, while the log-likelihood of the allele coverage is smaller for the optimal minor than for the true minor. However, we see that the decrease in the log-likelihood allele coverage is outweighed by an increase in the genotype probability.

3.1.3 Both profiles unknown

Lastly, we assumed that both the profiles were unknown. This is indicated by supplying an empty list (or just `NULL`) to the `knownProfiles` argument. Furthermore, we have increased the number of sub-populations, the population size, allowed for more outer iterations, and increased the termination counter `numberOfIterationsEqualMinMax`, giving the algorithm more times to search the space.

```
bothUnknown <- setHypothesis(sampleTibble, numberOfContributors, list(), theta)
```

```
## Warning: Unknown or uninitialised column: `Region`.
```

```
controlMultiplePopulation <-
  optimalUnknownProfileCombination.control(
    numberOfPopulations = 64,
    numberOfIterations = 100,
    populationSize = 300,
    numberOfFittestIndividuals = 1,
    numberOfIterationsEqualMinMax = 50,
    hillClimbingIterations = 0,
    mutationDecayRate = 1,
    parentSelectionWindowSize = 12,
    trace = FALSE,
    tolerance = rep(1e-8, 4)
  )

multiplePopulationBenchmarkUnknown <- microbenchmark(
  optimalMultipleUnknown <-
    optimalUnknownProfileCombination(sampleTibble = sampleTibble,
                                     markerImbalances = markerImbalances,
                                     H = bothUnknown[[1]],
                                     potentialParentsList = potentialParentsList,
                                     noiseParameters = noiseParameters,
                                     control = controlMultiplePopulation),
  times = 1)
```

The algorithm terminated in 2.178217×10^4 seconds. The parameters, log-likelihoods, and fitness of the optimal unknown genotype combination is:

```
optimalMultipleUnknown$U[[1]][c("Parameters", "LogLikelihoods", "Fitness")]
```

```
## $Parameters
## $Parameters$SampleParameters
## [1] 1412.999164    5.840008
##
```

```
## $Parameters$MixtureParameters
## [1] 0.7050434 0.2949566
##
## $Parameters$MarkerImbalanceParameters
##  [1] 1.0482923 0.7300408 0.5651653 1.4873667 0.9924879 1.2717018 1.5367861
##  [8] 1.1617728 0.7237483 0.4826379
##
## $Parameters$NoiseParameters
## [1] 3.0 2.0 0.5
##
##
## $LogLikelihoods
## [1] -276.68886  -32.98084    0.00000
##
## $Fitness
## [1] -309.6697
```

3.2 Approximating the LR

In our case of fictitious murder, we will assume that the victim is known and its profile typed. We will define the following competing hypotheses:

- The prosecution hypothesis, $\mathcal{H}_p$: The suspect, with genotype $\boldsymbol{g}_S$, is the perpetrator.
- The defence hypothesis, $\mathcal{H}_d$: Another random person from the population is the perpetrator.

In `R` we define a hypothesis by the `setHypothesis` function. It takes the arguments: `sampleTibble`, the total number of contributors to the mixture, `numberOfContributors`, a list of the known profiles to the mixture, `knownProfilesList`, and the inbreeding coefficient, `theta`.

```
theta = 0
knownProfilesHp <- trueProfiles
knownProfilesHd <- trueProfiles["V"]

Hp <- setHypothesis(sampleTibble, numberOfContributors, knownProfilesHp, theta)
Hd <- setHypothesis(sampleTibble, numberOfContributors, knownProfilesHd, theta)
```

An object of class `hypothesis` holds the total number of contributors, the number of known contributors, the combined genotype matrix of the known contributors, the inbreeding coefficient, and the allele frequencies of the chosen population.

Given the `sampleTibble` and the two competing hypotheses, we can initialise the EA for obtaining an approximate likelihood ratio by calling the `LR` function. As before we run the parallel algorithm with twelve sub-populations.

```
LRParallelPopulations <- LR(
  sampleTibble = sampleTibble,
  Hp = Hp,
  Hd = Hd,
  markerImbalances = markerImbalances,
  potentialParentsList = potentialParentsList,
  noiseParameters = noiseParameters,
  stutterRatioModel = stutterRatioModel,
  control = optimalUnknownProfileCombination.control(
    numberOfPopulations = 32, numberOfIterations = 150,
    populationSize = 75, numberOfFittestIndividuals = 1000,
    hillClimbingIterations = 0, mutationDecayRate = 1,
```

```
    parentSelectionWindowSize = 6, simplifiedReturn = FALSE,
    allowParentSurvival = TRUE, trace = FALSE,
    tolerance = rep(1e-8, 4))
)
```

```
## Warning in mclapply(seq_along(numberOfAlleles), function(m) {: all scheduled
## cores encountered errors in user code

## Warning in mclapply(seq_along(numberOfAlleles), function(m) {: all scheduled
## cores encountered errors in user code
```

The `hypothesis` class can take more than a single prosecutors or defence hypothesis (and, therefore, so can the `LR` function), and the `LR` function returns the likelihood ratios of every pairwise comparison of these hypotheses. The comparison table is accessed in the returned list as:

```
LRParallelPopulations$ComparisonTable
```

```
##   Hp Hd  Log10LR
## 1  2  2 15.52069
```

References

Balding, David J., and Richard, A. Nichols. 1994. "DNA profile match probability calculation: how to allow for population stratification, relatedness, database selection and single bands." *Forensic Science International* 64: 125–40.

Bleka, Øyvind., Geir Storvik, and Peter Gill. 2016. "EuroForMix: An open source software based on a continuous model to evaluate STR DNA profiles from a mixture of contributors with artefacts." *Forensic Science International: Genetics* 21: 35–44.

Cowell, Robert G., Therese Graversen, Steffen L. Lauritzen, and Julia Mortera. 2015. "Analysis of Forensic DNA Mixtures with Artefacts." *Royal Statistical Society. Journal Series C: Applied Statistics* 64: 1–32.

Steele, Christopher D., Matthew Greenhalgh, and David J. Balding. 2016. "Evaluation of low-template DNA profiles using peak heights." *Statistical Applications in Genetics and Molecular Biology* 15: 431–45.

Taylor, Duncan, Jo-Anne Bright, and John S. Buckleton. 2013. "The interpretation of single source and mixed DNA profile." *Forensic Science International: Genetics* 7: 516–28.

Tvedebrink, Torben, Maria Asplund, Poul Svante Eriksen, Helle Smidt Mogensen, and Niels Morling. 2013. "Estimating drop-out probabilities of STR alleles accounting for stutters, detection threshold truncation and degradation." *Forensic Science International: Genetics Supplement Series* 4: e51–e52.

Vilsen, Søren B., Torben Tvedebrink, Poul Svante Eriksen, Helle Smidt Mogensen, and Niels Morling. 2017. "Analysing allelic drop-out in MPS STR forensic genetics data."