Using the LikeLTD peak heights model

Christopher Steele, Adrian Timpson, Mayeul d'Avezac,
James Hetherington, David Balding

June 11, 2017

This vignette assumes familiarity with both programming in R and forensic DNA analysis, only demonstrating how to run an analysis. For a more in depth discussion of LikeLTD see `likeLTDguide.pdf` provided with the package.

1 Inputs

LikeLTD evaluates the weight-of-evidence (WoE) against a queried contributor (Q) in the form of a ratio of the likelihood of the evidence given the prosecution hypothesis (H_p) against that given the defence hypothesis (H_d) in which a random individual, X, replaces Q. To evaluate the WoE the user must supply a crime-stain profile (CSP), at least one reference profile (that of Q) and an allele frequency database for at least one population of interest. The reference profiles of any assumed contributors other than Q can also be provided. A number of allele frequency databases are supplied with LikeLTD, spanning the NGMSelect (`DNA17`), SGM+ (`SGMplus`) and Identifiler (`Identifiler`) STR profiling kits, but the user may specify their own database. Similarly, the case shown here uses example CSP and reference profile files supplied with LikeLTD, but the user may specify their own.

```
>   require(likeLTD)
>   # Case we are going to be evaluating
>   caseName = "Laboratory"
>   datapath = file.path(system.file("extdata", package="likeLTD"),
+       'laboratory')
>   # File paths and case name for allele report
>   admin = pack.admin.input.peaks(
+       peaksFile = file.path(datapath, 'laboratory-CSP.csv'),
+       refFile = file.path(datapath, 'laboratory-reference.csv'),
+       caseName = caseName,
+       detectionThresh = 20
+       )

>   # Generate allele report
>   allele.report.peaks(admin)
```

The important arguments to `pack.admin.input.peaks` are:

`peaksFile`: Path to CSP file in the same format as `laboratory-CSP.csv`. May contain multiple replicates.

`refFile`: Path to reference file in the same format as `laboratory-reference.csv`. May contain multiple reference profiles.

`databaseFile`: Path to database file. Only used if `kit=NULL`. The database file specified must contain a LUS column.

`kit`: One of `"DNA17"`, `"SGMplus"`, `"Identifiler"` or `NULL`. If both `databaseFile` and `kit` are `NULL`, the DNA17 database is used as default.

Designation	S	DS and OS
Non-alleleic	$x < 0.05$	$x < 0.05$
Uncertain	$0.05 \leq x < 0.15$	$0.05 \leq x < 0.1$
Allelic	$x \geq 0.15$	$x \geq 0.1$

Table 1: Determination of which peaks in stutter (S), over-stutter (OS) or double-stutter (DS) positions are treated as non-allelic, uncertain or allelic when estimating `nUnknowns`. x indicates the ratio of the stutter position peak height to the parent peak height.

`linkageFile`: File of pairwise recombination rates between loci. If `NULL` the default supplied with LikeLTD is used. This is only used when Q and X are assumed related.

`detectionThresh`: The detection threshold used to analyse the electrophoresis results. This can either be a single value that covers all loci, or a named list with a separate detection threshold for each locus. Defaults to 20 RFU.

The allele report generated by LikeLTD gives suggestions on the the number of unprofiled contributors to include in the analysis, and on whether or not to model dropin, as well as summarising the data. Note the suggestions of the number of unprofiled contributors and dropin are based on designations of allelic, uncertain and non-allelic peaks according to the criteria given in Table 1.

If instead you wish to specify a separate detection threshold at each locus, usually for a different threshold at each epg lane, the arguments to `pack.admin.input.peaks` should be:

```
> # Specifying locus specific detection thresholds
> admin = pack.admin.input.peaks(
+             peaksFile = file.path(datapath, 'laboratory-CSP.csv'),
+             refFile = file.path(datapath, 'laboratory-reference.csv'),
+             caseName = "Laboratory",
+             detectionThresh = list(D10S1248=20,vWA=20,D16S539=20,D2S1338=20, # blue
+                                    D8S1179=30,D21S11=30,D18S51=30, # green
+                                    D22S1045=40,D19S433=40,TH01=40,FGA=40, # black
+                                    D2S441=50,D3S1358=50,D1S1656=50,D12S391=50,SE33=50) # red
+             )
```

2 Hypothesis generation

Based on the suggestions of the allele report, or through manual inspection, the user must decide on a set of arguments to pass to LikeLTD in order for it to generate the prosecution and defence hypotheses.

```
>   # Enter arguments
>   args = list(
+         nUnknowns = 1,
+         doDropin = FALSE,
+         ethnic = "NDU1",
+         adj = 1,
+         fst = 0.03,
+         relationship = 0
+         )
>   # Create hypotheses
>   hypP = do.call(prosecution.hypothesis.peaks, append(admin,args))
>   hypD = do.call(defence.hypothesis.peaks, append(admin,args))
```

The arguments that may be handed to either `prosecution.hypothesis.peaks` or `defence.hypothesis.peaks` are:

`nUnknowns`: The number of unprofiled contributors to include under H_p. H_d automatically adds an unprofiled contributor, X, that replaces Q. The allele report suggests a value which must be either 0, 1 or 2. Defaults to 0.

`doDropin`: Logical, whether or not to model dropin. Suggested by allele report. Defaults to `FALSE`.

`ethnic`: Which database to use. Must match a column heading in the chosen database file. Defaults to `"NDU1"` (Caucasian). Other populations included in the DNA17 database are `"NDU1"` (Caucasian), `"NDU2"` (African + Afro-Caribbean), `"NDU3"` (South Asian), `"NDU4"` (East Asian), `"NDU6"` (African) and `"NDU7"` (Afro-Caribbean).

`adj`: Sampling adjustment. Defaults to 1.

`fst`: Fixation index, accounts for distant relatedness between Q and X. Defaults to 0.03.

`relationship`: Assumed relationship between Q and X. Can take values between 0 and 7:

0 Unrelated.

1 Parent/offspring.

2 Siblings.

3 Uncle (or aunt)/nephew (or niece).

4 Half-uncle (or half-aunt)/half-nephew (or half-niece).

5 Cousins.

6 Grandparent/grandchild.

7 Half-siblings.

The ordering of relationships does not alter the computation, so is unspecified e.g. if `relationship=1`, the relationship may be Q as parent and X as offspring, or Q as offspring and X as parent.

`doDoubleStutter`: Logical, whether or not to model double stutter. Defaults to `TRUE`.

`doOverStutter`: Logical, whether or not to model over stutter. Defaults to `TRUE`.

`combineRare`: Logical, whether or not to combine unobserved alleles into a single allele. Defaults to `TRUE`.

`rareThreshold`: If `combineRare=TRUE`, this parameter gives the population allele probability below which all unobserved alleles are combined. Defaults to 1 (all unobserved alleles).

3 Optimisation

Once the hypotheses have been generated, the next step is to create the likelihood functions along with their associated parameters for optimisation, and then to perform the optimisation.

```
>   # Generate likelihood functions and optimisation parameters
>   paramsP = optimisation.params.peaks(hypP,verbose=FALSE)
>   paramsD = optimisation.params.peaks(hypD,verbose=FALSE)
>   # reduce number of iterations for demonstration purposes
>   paramsP$control$itermax=25
>   paramsD$control$itermax=25

>   # Run optimisation
>   results = evaluate.peaks(paramsP, paramsD, n.steps=1,
+       converge=FALSE)
```

Here `n.steps` and `converge` are specified to reduce run-time for demonstration purposes; if unspecified `evaluate.peaks` determines the number of steps after the first step has completed. Other parameters the user may wish to pass to `evaluate.peaks` include:

interim: Logical, whether or not to generate interim reports after every step of optimisation. Defaults to TRUE.

seed.input: Integer specification of a seed to use before optimisation. If unspecified, LikeLTD uses a numeric representation of the current time, date and process ID at the time of running `evaluate`.

The behaviour of the optimisation algorithm may be modified by specifying `tolerance`, `CR.start`, `CR.end` and `nConverged`, although we do not recommend altering these parameters unless you understand what effect each of the parameters has.

The `results` object contains the WoE at each step of optimisation (`WoE`), the prosecution and defence $-\log_{10}$ likelihoods at each step of optimisation (`Lp` and `Ld`), the prosecution and defence objects returned by DEoptim after optimisation (`Pros` and `Def`) and information about the seed that was used (`seed.used` and `seed.input`).

4 Generate output report

Once optimisation has been completed, an output report summarising the results can be generated. This outputs the WoE against Q, likelihoods at each locus, the inverse match probability (IMP, the theoretical maximum WoE), DNA contribution estimates for each contributor in each replicate, degradation estimates for each contributor and dropin estimates if dropin is modelled.

```
>   # Generate output report
>   output.report.peaks(hypP,hypD,results)
```

5 Likely genotypes of the unknown contributors

Following optimisation it is possible to return the most likely marginal genotypes for each contributor or joint genotypes of all contributors under either H_p or H_d. These can be useful for searching a national database with one of the unknown genotypes.

```
>   # Get the most likely single-contributor genotypes
>   gensMarginal = get.likely.genotypes.peaks(hypD,paramsD,
+       results$Def)
>   # Return joint genotypes and probabilities
>   gensJoint = get.likely.genotypes.peaks(hypD,paramsD,
+       results$Def,joint=TRUE)
```

It is also possible to return the probabilities for all of the genotype combinations under the specified hypothesis. This has been used to compute the WoE against a degraded incomplete reference profile, drawing incomplete genotypes from the posterior probability for which any known alleles match, but may have other uses.

```
>   # Get the posterior likelihoods for all genotype combinations
>   gensPosterior = get.likely.genotypes.peaks(hypD,paramsD,
+       results$Def,posterior=TRUE)
```

6 Diagnostic

A visual inspection of the fit of the optimised parameters to the supplied CSP data can be performed using the `peaks.results.plot` function, which plots the 95% equal-tailed probability interval of the gamma distribution given the optimised parameters, assuming the joint genotype combination that is most likely (see Figure 1).

```
>   # Plot CSP with most likely genotypes
>   peaks.results.plot(hypD,results$Def,replicate=1)
```

```
[1] "Proportion of peaks within 95% probability interval: 0.942528735632184"
[1] "Proportion of peaks within 50% probability interval: 0.505747126436782"
```

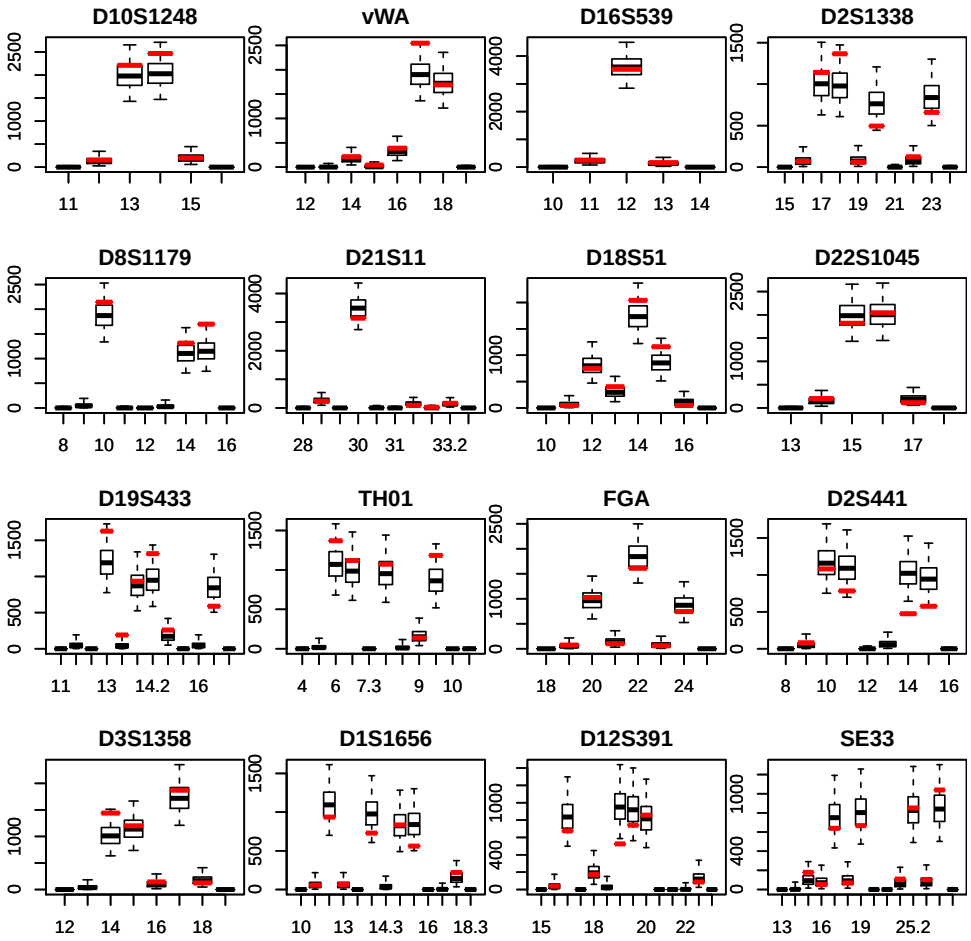

Figure 1: Boxes show the central 50% (inter-quartile range) of the gamma distribution for each hypothesised peak, whiskers represent the 95% equal-tailed probability interval and red bars show observed peak heights. RFU is displayed on the y-axes while allele labels corresponding to boxplots are displayed on the x-axes.