Package 'likeLTD'

October 24, 2016

Title Tools to Evaluate DNA Profile Evidence

Description Tools to determine DNA profile Weight of Evidence.
For further information see the 'likeLTD' guide provided,
or Balding, D.J. (2013) <DOI:10.1073/pnas.1219739110>.

Depends R (>= 2.10), DEoptim, ggplot2, gtools, rtf

Suggests svUnit, scales

Imports gdata, tools, tcltk

Version 6.1.1

Date 2016-10-24

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

Maintainer Christopher Steele <c.steele.11@ucl.ac.uk>

License GPL-3

URL https://sites.google.com/site/baldingstatisticalgenetics/

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-10-24 23:27:19

R topics documented:

`allele.report` *likeLTD::allele.report*

Description

Outputs a docx summary of the data inputs, to assist the user in choosing parameter inputs for a full evaluation

Usage

```
allele.report(admin,file=NULL)
```

Arguments

admin	List containing administration data, as packed by pack.admin.input()
file	A file name for the allele report. By default a sequential filename is created to avoid accidental overwriting.

Details

The allele report summarises alleles present in both the crime scene profile and the reference profiles, prior to the main statistical evaluation. Parameter inputs for nUnknowns and doDropin are suggested. The report is placed in the directory specified by admin$outputPath.

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# Next we generate an allele report
allele.report(admin)
## End(Not run)
```

---

allele.report.peaks *likeLTD::allele.report.peaks*

Description

Outputs a docx summary of the data inputs, to assist the user in choosing parameter inputs for a full evaluation

Usage

```
allele.report.peaks(admin,file=NULL, figRes=300, dropinThresh=3)
```

Arguments

admin	List containing administration data for peak heights inputs, as packed by pack.admin.input.peaks()
file	A file name for the allele report. By default a sequential filename is created to avoid accidental overwriting.
figRes	Resolution of figures in the allele report. Defaults to 300 DPI.
dropinThresh	If the estimated DNA contribution of Q is more than dropinThresh times the estimated DNA contribution of any unknown (estimated by k-means) then it will be suggested that that unknown contributor could be modelled as dropin instead.

Details

The allele report summarises the crime scene profile and the reference profiles, prior to the main statistical evaluation. Parameter inputs for nUnknowns and doDropin are suggested. The report is placed in the directory specified by admin$outputPath.

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# generate allele report
allele.report.peaks(admin)
## End(Not run)
```

---

compatible.genotypes *likeLTD::compatible.genotypes*

Description

Genetics make-up of nUnknown unprofiled contributors for given locus.

Usage

```
compatible.genotypes(cspPresence, profPresence, alleleNames, nUnknowns,
                             dropin=FALSE, missingReps=NULL)
```

Arguments

cspPresence	Crime Scene Profile for given locus. This should be a matrix where rows are replicas and contributors, columns are alleles for the given locus, and the input is TRUE or FALSE and indicates the presence of that particular allele.
profPresence	Known profiles for given locus. Same type of format as cspLocus.
alleleNames	Name of the alleles, e.g. columns of the previous two
nUnknowns	number of unknown contributors
dropin	TRUE if modelling drop-ins
missingReps	List of booleans indicating whether a replicate is missing in cspPresence

Value

A m by (2n) matrix where the colums (grouped by twos) correspond to contributors, and each row is their potential contribution to the CSP.

---

create.likelihood *likeLTD::create.likelihood*

Description

Creates an objective function

Usage

```
create.likelihood(hypothesis, addAttr=FALSE, ...)
```

Arguments

hypothesis	Hypothesis for which to create the objective function
addAttr	Whether to add attributes to the objective functions. There are two attributes: (1) "hypothesis" referencing the input hypothesis, (ii) 'functions' containing the individual objective functions per Locus. The latter contain further attributes.
...	Any named parameter to modify the hypothesis, e.g. nUnknowns

Details

The objective function is created from the hypothesis. Itself, it takes as arguments the nuisance parameters and, optionally, the parameters for the penalty function. This particular flavor of the objective returns the product of the likelihood and penalties across all loci.

Value

A function

See Also

create.likelihood.vectors, create.likelihood.log, penalties, Objective Functions

`create.likelihood.log` *likeLTD::create.likelihood.log*

Description

Creates an objective function

Usage

```
create.likelihood.log(hypothesis, addAttr=FALSE, ...)
```

Arguments

`hypothesis`	Hypothesis for which to create the objective function
`addAttr`	Whether to add attributes to the objective functions. There are two attributes: (1) "hypothesis" referencing the input hypothesis, (ii) 'functions' containing the individual objective functions per Locus. The latter contain further attributes.
`...`	Any named parameter to modify the hypothesis, e.g. `nUnknowns`

Details

The objective function is created from the hypothesis. Itself, it takes as arguments the nuisance parameters and, optionally, the parameters for the penalty function. This particular flavor of the objective returns the sum of the log-likelihood and the sum of log-penalties across loci.

Value

A function

See Also

create.likelyhood.vectors, create.likelihood, penalties, Objective Functions

---

`create.likelihood.vectors`
likeLTD::create.likelihood.vectors

Description

Creates an objective function

Usage

```
create.likelihood.vectors(hypothesis, addAttr=FALSE, likeMatrix=FALSE, ...)
```

Arguments

hypothesis	Hypothesis for which to create the objective function
addAttr	Whether to add attributes to the objective functions. There are two attributes: (1) "hypothesis" referencing the input hypothesis, (ii) 'functions' containing the individual objective functions per Locus. The latter contain further attributes.
likeMatrix	Whether to return likelihoods for every genotype combination, or a likelihood summed over all genotypes after optimisation. Set to TRUE for individual genotype likelihoods. This is used for `get.likely.genotypes`.
...	Any named parameter to modify the hypothesis, e.g. `nUnknowns`

Details

The objective function is created from the hypothesis. Itself, it takes as arguments the nuisance parameters and, optionally, the parameters for the penalty function. This particular flavor of the objective returns a list containing two items: (i) the likelihoods per locus, (ii) the penalties per locus.

Value

A function

See Also

create.likelihood, create.likelihood.log, get.likely.genotypes, penalties, Objective Functions

---

`create.likelihood.vectors.peaks`

likeLTD::create.likelihood.vectors.peaks

Description

Creates an objective function for peak height data

Usage

```
create.likelihood.vectors.peaks(hypothesis, addAttr=FALSE,
likeMatrix=FALSE, diagnose=FALSE,...)
```

Arguments

hypothesis	Hypothesis for which to create the objective function
addAttr	Whether to add attributes to the objective functions. There are two attributes: (1) "hypothesis" referencing the input hypothesis, (ii) 'functions' containing the individual objective functions per Locus. The latter contain further attributes.

likeMatrix	Whether to return likelihoods for every genotype combination, or a likelihood summed over all genotypes after optimisation. Set to TRUE for individual genotype likelihoods. This is used for get.likely.genotypes.peaks.
diagnose	Logical. If TRUE returns a list of peak heights, estimated mean peak height and peak height standard deviation for each genotype combination. This can be used to obtain Z values for each fitted peak.
...	Any named parameter to modify the hypothesis, e.g. nUnknowns

Details

The objective function is created from the hypothesis. Itself, it takes as arguments the nuisance parameters and, optionally, the parameters for the penalty function. This particular flavor of the objective returns a list containing two items: (i) the likelihoods per locus, (ii) the penalties per locus.

Value

A function

See Also

get.likely.genotypes.peaks, penalties.peaks, Objective Functions

---

CSP.heights.plot *likeLTD::CSP.heights.plot*

Description

Plot one replicate from a peak heights CSP.

Usage

```
CSP.heights.plot(csp, refs, dbFile = NULL, kit = NULL,
outputFile = NULL, toPlot = NULL,
detectThresh = NULL, uncThresh = 0.05,
stutterThresh = 0.15, doStutter = FALSE,
replicate = 1,...)
```

Arguments

csp	CSP as returned from read.peaks.profile.
refs	Reference profiles as returned by read.known.profiles.
dbFile	Path to allele database file. If NULL, kit is used instead.
kit	Choice of an allele database that is included wiht likeLTD. If both dbFile and kit are NULL, the DNA17 database is used.
outputFile	Path to output file. If NULL, the plot is output to a plot window instead.

toPlot	An integer vector specifying which loci to plot. If NULL, all loci are plotted.
detectThresh	Detection threshold used for electrophoresis analysis. If NULL, this is not plotted.
uncThresh	Threshold for suggesting that a peak is uncertain.
stutterThresh	Threshold for suggesting a peak is allelic.
doStutter	Logical. Specifies whether to suggest peaks as non-allelic/uncertain/allelic.
replicate	Which replicate to plot.
...	Extra parameters to pass to plot.

Details

If refFile is specified, the plotted peaks are coloured according to which reference individuals possess that allele. If detectThresh is specified, a horizontal line is plotted indicating the threshold. If doStutter=TRUE, the labels for each peak are coloured by a crude estimation of which peaks are allelic, uncertain, or non-allelic.

Value

Pdf file or plot window.

See Also

allele.report.peaks

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# plot CSP heights
CSP.heights.plot(csp=read.peaks.profile(admin$peaksFile),
ref=read.known.profiles(admin$refFile))

# to plot just the first four loci
CSP.heights.plot(csp=read.peaks.profile(admin$peaksFile),
ref=read.known.profiles(admin$refFile),
toplot=1:4)

# to estimate which peaks are non-allelic
CSP.heights.plot(csp=read.peaks.profile(admin$peaksFile),
ref=read.known.profiles(admin$refFile),
doStutter=TRUE)
```

```
# to display detection threshold
CSP.heights.plot(csp=read.peaks.profile(admin$peaksFile),
ref=read.known.profiles(admin$refFile),
detectThresh=20)

## End(Not run)
```

`defence.hypothesis` *likeLTD::defence.hypothesis*

Description

Helper function to create the input for the defence.

Usage

```
defence.hypothesis(cspFile, refFile, ethnic='NDU1', nUnknowns=0,
                        adj=1e0, fst=0.02, databaseFile=NULL,
                        linkageFile=NULL,
                        doDropin=FALSE,
                        combineRare=TRUE, rareThreshold=0.05,
                        kit=NULL, relationship=0, ...)
```

Arguments

`cspFile`	Path to the crime scene profile.
`refFile`	Path to the known profiles.
`ethnic`	Ethnicity, e.g. subgroup within the allele frequency database.
`nUnknowns`	Number of unknown contributors for which to perform calculation.
`adj`	Allele frequency adjustment parameter.
`fst`	Allele fraction adjustment F_{ST}.
`databaseFile`	Path to the allele database. If NULL, then defaults to the NGMSelect database provided with `likeLTD`.
`linkageFile`	Path to recombination rate information. If NULL then defaults to the linkage file provided with `likeLTD`. Only used if `relatedness=c(0.5,0.25)` i.e. when Q and X are siblings.
`relationship`	Specified relationship between Q and X. Can take values of 0=unrelated, 1=parent/offspring, 2=siblings, 3=uncle/nephew, 4=half-uncle/half-nephew, 5=cousins, 6=grandparent/grandchild, 7=half-siblings.
`doDropin`	Whether or not to model drop-in.
`combineRare`	Whether or not to combine rare unobserved alleles into a single allele.
`rareThreshold`	If `combineRare=TRUE`, this is the probability threshold below which an allele is classed as rare, and therefore combined with other rare unobserved alleles.

`kit`	Parameter specifying which allele database supplied with likeLTD to use if `linkageFile` is not specified. Possibilities are "DNA17", "Identifiler" and "SGMplus".
`...`	Other parameters to determine how to perform calculations.

Details

It loads the CSP, known profiles, and allele database from file. It removes the queried individual from the known profiles. It increments the number of unknown contributors by one (to make up for the queried individual). The reference individual is set to "X" (first unprofiled contributor) by default.

Value

A list of named input parameters, or hypothesis, suitable for the defence.

See Also

prosecution.hypothesis

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# Enter arguments
args = list(
        nUnknowns = 1,
        doDropin = FALSE,
        ethnic = "EA1",
        adj = 1,
        fst = 0.02,
        relatedness = c(0,0)
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis, append(admin,args))
hypD = do.call(defence.hypothesis, append(admin,args))

## End(Not run)
```

`defence.hypothesis.peaks` *likeLTD::defence.hypothesis.peaks*

Description

Helper function to create the input for the defence using peak height data.

Usage

```
defence.hypothesis.peaks(peaksFile, refFile, ethnic='NDU1', nUnknowns=0,
                         adj=1e0, fst=0.03, databaseFile=NULL,
                         linkageFile=NULL,
                         detectionThresh=20, doDropin=FALSE,
                         doDoubleStutter=TRUE,doOverStutter=TRUE,
                         combineRare=TRUE, rareThreshold=1,
                         kit=NULL, relationship=0, ...)
```

Arguments

`peaksFile`	Path to the crime scene profile.
`refFile`	Path to the known profiles.
`ethnic`	Ethnicity, e.g. subgroup within the allele frequency database.
`nUnknowns`	Number of unknown contributors for which to perform calculation.
`adj`	Allele frequency adjustment parameter.
`fst`	Allele fraction adjustment F_{ST}.
`databaseFile`	Path to the allele database. If NULL, then defaults to the NGMSelect database provided with `likeLTD`. Must include longest uninterrupted sequence (LUS) values for alleles.
`linkageFile`	Path to recombination rate information. If NULL then defaults to the linkage file provided with `likeLTD`. Only used if `relationship` is not 0 or 1 i.e. when Q and X are closely related but not parent/offspring.
`relationship`	Specified relationship between Q and X. Can take values of 0=unrelated, 1=parent/offspring, 2=siblings, 3=uncle/nephew, 4=half-uncle/half-nephew, 5=cousins, 6=grandparent/grandchild, 7=half-siblings.
`detectionThresh`	Detection threshold for peaks. Can be a single value, or a named list containing one value per locus.
`doDropin`	Whether or not to model drop-in. Note dropin is not currently possible with the peak heights model.
`doDoubleStutter`	Logical. Whether or not to model double stutter.
`doOverStutter`	Logical. Whether or not to model over stutter.

combineRare	Whether or not to combine rare unobserved alleles into a single allele.
rareThreshold	If combineRare=TRUE, this is the probability threshold below which an allele is classed as rare, and therefore combined with other rare unobserved alleles.
kit	Parameter specifying which allele database supplied with likeLTD to use if linkageFile is not specified. Possibilities are "DNA17-lus".
...	Other parameters to determine how to perform calculations.

Details

It loads the CSP, known profiles, and allele database from file. It removes the queried individual from the known profiles. It increments the number of unknown contributors by one (to make up for the queried individual).

Value

A list of named input parameters, or hypothesis, suitable for the defence.

See Also

prosecution.hypothesis.peaks

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# Enter arguments
args = list(
        nUnknowns = 1
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis.peaks, append(admin,args))
hypD = do.call(defence.hypothesis.peaks, append(admin,args))

## End(Not run)
```

DEoptimLoop *likeLTD::DEoptimLoop*

Description

Values returned by DEoptim after optimization. See DEoptim help page for more details.

Usage

```
DEoptimLoop(PARAMS, tolerance=1e-6)
```

Arguments

PARAMS	Parameters for optimization as generated by optimization.params
tolerance	If the relative difference between the current result, and the last checked result is less than this value, then it is classed as converged.

Details

Optimize over parameter space, while checking for converence every 50 iterations.

Value

List.

See Also

DEoptim

determine.dropout *determine.dropout*

Description

An individual is subject to dropout if the individual's allele at one or more locus is not in the crime-scene profile.

Usage

```
determine.dropout(knownProfiles, cspProfile)
```

Arguments

knownProfiles	A matrix containing the profiles to examin.
cspProfile	A matrix containing the crime scene profile.

Value

A list with one element per individual. Each is True if that individual is subject to dropout.

See Also

read.csp.profile, read.known.profiles

DNA17-db *NGMSelect allele database*

Description

A table with allele frequencies for NGMSelect loci. Populations are "NDU1" (Caucasian), "NDU2" (African + Afro-Caribbean), "NDU3" (South Asian), "NDU4" (East Asian), "NDU6" (African), "NDU7" (Afro-Caribbean).

Format

A table, as described in `load.allele.database`.

---

evaluate *likeLTD::evaluate*

Description

Optimize both prosecution and defence likelihoods, returning the weight of evidence.

Usage

```
evaluate(P.pars, D.pars, tolerance=1e-5, n.steps=NULL, progBar=TRUE, interim=TRUE,
CR.start=0.1, CR.end=0.7, seed.input=NULL)
```

Arguments

`P.pars`	Parameters for prosecution hypothesis, as generated by optimisation.params.
`D.pars`	Parameters for prosecution hypothesis, as generated by optimisation.params.
`tolerance`	Tolerance for the final chunk of optimisation. If the relative difference between the current result, and the last checked result is less than this value, then it is classed as converged.
`n.steps`	Number of steps to run. Defaults to NULL. if n.steps is NULL , the number of steps to run is determined by the mean of the standard deviation of the initial phase of the intial chunk of optimisation for prosecution and defence.
`progBar`	Logical, stating whether to display a graphical progress basr or not. This should be set to FALSE is the user does not have graphical capabilities e.g. if running from command line on a server.

interim	Logical, stating whether or not to generate interim reports. If set to TRUE a basic set of results after each step is output to "Interim.csv", and an image of the content of evaluate() or evaluate.from.interim() "interim.RData" are both stored in the current working directory. The latter can be used by evaluate.from.interim() to continue an evaluation from its previous state. Each step will write over the results from the previous step.
CR.start	Numerical, between 0 and 1, used by DEoptim as CR argument, at the start of the search. Gradually moves towards CR.end to allow a broader initial search, gradually becoming more localised in parameter space. See DEoptim for further details.
CR.end	Numerical, between 0 and 1,see details for CR.start.
seed.input	An integer that should be specified if the user wishes to set a particular seed. If not specified, the program sets the seed to an integer representation of the present time, date and process ID.

Details

Optimize over parameter space, using a geometric progression of crossover rate and tolerance. Both prosecution and defence cases are optimized simultaneously.

Value

A list containing five elements:

Pros	Prosecution results, structured as results from DEoptim::DEoptim.
Def	Defence results, structured as results from DEoptim::DEoptim.
WoE	WoE for each chunk. The final value if the final WoE.
seed.used	Seed that is set at the beginning of computation.
seed.input	Seed that is input by the user.

See Also

DEoptim,DEoptimLoop

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# Enter arguments
```

```
args = list(
        nUnknowns = 1,
        doDropin = FALSE,
        ethnic = "EA1",
        adj = 1,
        fst = 0.02,
        relatedness = c(0,0)
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis, append(admin,args))
hypD = do.call(defence.hypothesis, append(admin,args))

# Get parameters for optimisation
paramsP = optimisation.params(hypP)
paramsD = optimisation.params(hypD)

# Run optimisation
# n.steps set for demonstration purposes
results = evaluate(paramsP, paramsD, n.steps=1)

## End(Not run)
```

`evaluate.from.interim` *likeLTD::evaluate.from.interim*

Description

Optimize both prosecution and defence likelihoods, returning the weight of evidence.

Usage

```
evaluate.from.interim(file)
```

Arguments

`file`	Restricted to only accept a filepath including the name "interim.RData", which is generated by evaluate() only if interim=TRUE. "interim.RData" will also be subsequently generated by evaluate.from.interim() thereafter, at regular intervals during the evaluation

Details

Optimize over parameter space, using a geometric progression of crossover rate and tolerance. Both prosecution and defence cases are optimized simultaneously.

Value

A list containing five elements:

`Pros`	Prosecution results, structured as results from DEoptim::DEoptim.
`Def`	Defence results, structured as results from DEoptim::DEoptim.
`WoE`	WoE for each chunk. The final value is the final WoE.
`seed.used`	Seed that is set at the beginning of computation; this will no longer apply if evaluate.from.interim.peaks is used.
`seed.input`	Seed that is input by the user; this will no longer apply if evaluate.from.interim.peaks is used.

See Also

evaluate

`evaluate.from.interim.peaks`
likeLTD::evaluate.from.interim.peaks

Description

Optimize both prosecution and defence likelihoods, returning the weight of evidence.

Usage

```
evaluate.from.interim.peaks(file)
```

Arguments

`file`	Restricted to only accept a filepath including the name "interim.RData", which is generated by evaluate.peaks() only if interim=TRUE. "interim.RData" will also be subsequently generated by evaluate.from.interim.peaks() thereafter, at regular intervals during the evaluation

Details

Optimize over parameter space, using a geometric progression of crossover rate and tolerance. Both prosecution and defence cases are optimized simultaneously.

Value

A list containing seven elements:

Pros	Prosecution results, structured as results from DEoptim::DEoptim.
Def	Defence results, structured as results from DEoptim::DEoptim.
WoE	WoE for each chunk. The final value is the final WoE.
Lp	-log_10 prosecution likelihood at each step usually.
Ld	-log_10 defence likelihood at each step usually.
seed.used	Seed that is set at the beginning of computation; this will no longer apply if evaluate.from.interim.peaks is used.
seed.input	Seed that is input by the user; this will no longer apply if evaluate.from.interim.peaks is used.

See Also

evaluate

---

evaluate.peaks *likeLTD::evaluate.peaks*

Description

Optimize both prosecution and defence likelihoods, returning the weight of evidence for peak height data.

Usage

```
evaluate.peaks(P.pars, D.pars, tolerance=1e-6, n.steps=NULL, interim=TRUE,
CR.start=0.1, CR.end=0.7, seed.input=NULL, converge=TRUE, nConverged=4, stepsFun=NULL)
```

Arguments

P.pars	Parameters for prosecution hypothesis, as generated by optimisation.params.peaks.
D.pars	Parameters for prosecution hypothesis, as generated by optimisation.params.peaks.
tolerance	Tolerance for optimisation. Optimisation continues until the last nConverged chunks have a relative difference between the current result and them less than this value.
n.steps	Number of steps to run. Defaults to NULL. if n.steps is NULL , the number of steps to run is determined by the mean of the standard deviation of the initial phase of the intial chunk of optimisation for prosecution and defence.
interim	Logical, stating whether or not to generate interim reports. If set to TRUE a basic set of results after each step is output to "Interim.csv", and an image of the content of evaluate() or evaluate.from.interim() "interim.RData" are both stored in the current working directory. The latter can be used by evaluate.from.interim() to continue an evaluation from its previous state. Each step will write over the results from the previous step.

CR.start	Numerical, between 0 and 1, used by DEoptim as CR argument, at the start of the search. Gradually moves towards CR.end to allow a broader initial search, gradually becoming more localised in parameter space. See DEoptim for further details.
CR.end	Numerical, between 0 and 1,see details for CR.start.
seed.input	An integer that should be specified if the user wishes to set a particular seed. If not specified, the program sets the seed to an integer representation of the present time, date and process ID.
converge	Logical. Whether or not to check for convergence of the last nConverged steps to the global best values.
nConverged	The number of steps with relitive difference to the current results less than tolerance that determines convergence.
stepsFun	Custom function to determine the number of steps to run after the first. Should have arguments sdStep, the standard deviation of the first step, nCont, the number of hypothesised contributors, and nReps, the number of replicates in the CSP.

Details

Optimize over parameter space, using a geometric progression of crossover rate and tolerance. The defence hypothesis is optimised before the prosecution hypothesis, with the result used to constrain some of the parameters of the prosecution optiimisation.

Value

A list containing seven elements:

Pros	Prosecution results, structured as results from DEoptim::DEoptim.
Def	Defence results, structured as results from DEoptim::DEoptim.
WoE	WoE for each chunk. The final value is the final WoE.
Lp	-log_10 prosecution likelihood at each step usually.
Ld	-log_10 defence likelihood at each step usually.
seed.used	Seed that is set at the beginning of computation.
seed.input	Seed that is input by the user.
runtime	List of three objects: elapsed, start, end. runtime$start and runtime$end are obtained through Sys.time at the beginning and end of computation, runtime$elapsed is obtained by difftime on runtime$start and runtime$end.

See Also

DEoptim,DEoptimLoop

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# Enter arguments
args = list(
        nUnknowns = 1
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis.peaks, append(admin,args))
hypD = do.call(defence.hypothesis.peaks, append(admin,args))

# Get parameters for optimisation
paramsP = optimisation.params.peaks(hypP)
paramsD = optimisation.params.peaks(hypD)

# reduce number of iterations for demonstration purposes
paramsP$control$itermax=25
paramsD$control$itermax=25

# Run optimisation
# n.steps and converge set for demonstration purposes
results = evaluate.peaks(paramsP, paramsD, n.steps=1,
      converge=FALSE)

## End(Not run)
```

`get.likely.genotypes` *likeLTD::get.likely.genotypes*

Description

Creates a list of the most likely genotypes at each locus, and the most likely whole-profile genotype.

Usage

```
get.likely.genotypes(hypothesis,params,results,
posterior=FALSE,joint=FALSE,prob=ifelse(joint==FALSE,0.1,0.05))
```

Arguments

hypothesis	Hypothesis object created by either defence.hypothesis or prosecution.hypothesis that was used for optimisation with DEoptimLoop.
params	Parameters object created by optimisation.params that was used for optimisation with DEoptimLoop.
results	Results object returned by DEoptimLoop
posterior	Logical indicating whether to return all genotype probabilities, rather than just the most likely.
joint	Logical indicating whether or not to return joint genotypes and probabilities. If FALSE, marginal genotypes and probabilities are returned instead.
prob	Probability threshold for single-locus genotype probabilities. Defaults to 0.1 if returning marginal probabilities, and 0.05 if returning joint probabilities.

Details

Either joint or marginal genotypes and genotype probabilities are given. Locus-specific genotypes are only given if their probabilty exceeds prob. The most likely whole-profile genotype is given, regardless of the probability threshold at each locus. Joint probabilities give the probability of a multi-contributor genotype, whereas marginal probabilities give the probability of a single contributor, summing over all the possible genotypes for all other contributors.

Value

locusSpecific	Locus genotypes and probabilities which are greater than prob for each contributor of joint=FALSE, or for all contributors if joint=TRUE.
topGenotype	Most likely whole-profile genotype for all contributors if joint=TRUE or for each contributor separately if joint=FALSE. The probability of each genotype is also given.

See Also

defence.hypothesis, prosecution.hypothesis, optimisation.params,DEoptimLoop

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# Enter arguments
```

```
args = list(
        nUnknowns = 1,
        doDropin = FALSE,
        ethnic = "EA1",
        adj = 1,
        fst = 0.02,
        relatedness = c(0,0)
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis, append(admin,args))
hypD = do.call(defence.hypothesis, append(admin,args))

# Get parameters for optimisation
paramsP = optimisation.params(hypP)
paramsD = optimisation.params(hypD)

# Run optimisation
# n.steps set for demonstration purposes
results = evaluate(paramsP, paramsD, n.steps=1)

# get most likely marginal genotypes under defence
get.likely.genotypes(hypD,paramsD,results$Def)

# get most likely joint genotypes under defence
get.likely.genotypes(hypD,paramsD,results$Def,joint=TRUE)

# get full posterior likelihoods
get.likely.genotypes(hypD,paramsD,results$Def,posterior=TRUE)

## End(Not run)
```

`get.likely.genotypes.peaks` *likeLTD::get.likely.genotypes.peaks*

Description

Creates a list of the most likely genotypes at each locus, and the most likely whole-profile genotype for peak height data.

Usage

```
get.likely.genotypes.peaks(hypothesis,params,results,
posterior=FALSE,joint=FALSE,prob=ifelse(joint==FALSE,0.1,0.05))
```

Arguments

`hypothesis`	Hypothesis object created by either `defence.hypothesis.peaks` or `prosecution.hypothesis.peaks` that was used for optimisation with `evaluate.peaks`.

params	Parameters object created by optimisation.params.peaks that was used for optimisation with evaluate.
results	Either prosecution or defence results returned by evaluate.peaks e.g. results$Pros or results$Def.
posterior	Logical indicating whether to return all genotype probabilities, rather than just the most likely.
joint	Logical indicating whether or not to return joint genotypes and probabilities. If FALSE, marginal genotypes and probabilities are returned instead.
prob	Probability threshold for single-locus genotype probabilities. Defaults to 0.1 if returning marginal probabilities, and 0.05 if returning joint probabilities.

Details

Either joint or marginal genotypes and genotype probabilities are given. Locus-specific genotypes are only given if their probabilty exceeds prob. The most likely whole-profile genotype is given, regardless of the probability threshold at each locus. Joint probabilities give the probability of a multi-contributor genotype, whereas marginal probabilities give the probability of a single contributor, summing over all the possible genotypes for all other contributors.

Value

locusSpecific	Locus genotypes and probabilities which are greater than prob for each contributor of joint=FALSE, or for all contributors if joint=TRUE.
topGenotype	Most likely whole-profile genotype for all contributors if joint=TRUE or for each contributor separately if joint=FALSE. The probability of each genotype is also given.

See Also

defence.hypothesis.peaks, prosecution.hypothesis.peaks, optimisation.params.peaks,evaluate.peaks

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# Enter arguments
args = list(
        nUnknowns = 1
        )
```

```
# Create hypotheses
hypP = do.call(prosecution.hypothesis.peaks, append(admin,args))
hypD = do.call(defence.hypothesis.peaks, append(admin,args))

# Get parameters for optimisation
paramsP = optimisation.params.peaks(hypP)
paramsD = optimisation.params.peaks(hypD)

# reduce number of iterations for demonstration purposes
paramsP$control$itermax=25
paramsD$control$itermax=25

# Run optimisation
# n.steps and converge set for demonstration purposes
results = evaluate.peaks(paramsP, paramsD, n.steps=1,
      converge=FALSE)

# get most likely marginal genotypes under defence
get.likely.genotypes.peaks(hypD,paramsD,
      results$Def)

# get most likely joint genotypes under defence
gensJoint = get.likely.genotypes.peaks(hypD,paramsD,
      results$Def,joint=TRUE)

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

## End(Not run)
```

`Identifiler-db` *Identifiler allele database*

Description

A table with allele frequencies for Identifiler loci. Populations are "EA1" (Caucasian), "EA3" (African/Afro-Caribbean), "EA4" (South Asian).

Format

A table, as described in `load.allele.database`.

initial.arguments *likeLTD::initial.arguments*

Description

Provides starting point values for optimization.

Usage

```
initial.arguments(hypothesis, ...)
```

Arguments

hypothesis	Hypothesis from which objective function was obtained.
...	Any parameter to modify the hypothesis.

Value

rcont	Vector of relative contributions. The relative contribution of the first known profile is set to 1 by default and thus not present in this vector.
degradation	Degradation for each contributor
locusAdjustment	Adjustment for each locus
power	Tvedebrink exponent
dropin	Dropin rate. Can be ignored if not dropin.
dropout	Dropout rates per replicate

See Also

optimisation.params

lgc-allele-freqs-wbp *Allele database*

Description

A table with allele frequencies.

Format

A table, as described in load.allele.database.

likeLTD *likeLTD*

Description

A package to compute likelihoods for low template DNA profiles.

Details

likeLTD ("likelihoods for Low Template DNA profiles") is an R package for computing likelihoods for DNA profiles. It is freely available under a GNU public license.

It is particularly suited for low-template and/or degraded DNA when alleles from some contributors may be subject to dropout. It can handle any number of profiled possible contributors and up to three unprofiled contributors.

likeLTD is currently in beta, please report any issues to the maintainer.

See https://sites.google.com/site/baldingstatisticalgenetics/ for an in-depth guide to **likeLTD**.

References

David J. Balding, "Evaluation of mixed-source, low-template dna profiles in forensic science", *Proceedings of the National Academy of Sciences of USA*, (2009)

linkage *Recombination rates for linked loci*

Description

A table recombination rates for a set of linked loci.

Format

An nxn table, where n is the number of loci in the database. If locus x and y are linked, then table[x,y] and table[y,x] contain the recombination rate for those loci. Unlinked loci have value NA.

`load.allele.database` *likeLTD::load.allele.database*

Description

Reads allele database from file, or loads database provided with **likeLTD**.

Usage

```
load.allele.database(path=NULL, kit=NULL)
```

Arguments

`path`	Path to the crime scene profile, or `NULL`. If `NULL`, then returns the allele database packaged with **likeLTD**.
`kit`	Only used if `path=NULL`. Specifies one of the allele database supplied with `likeLTD` to use. Possibilities are "DNA17", "Identifiler" and "SGMplus". Defaults to "DNA17" if both `path=NULL` and `kit=NULL`.

Details

If not NULL, then the input should consist of a file in the following format is using discrete model:

Marker	Allele	BP	EA1	EA2	EA3	...
TH01	5	166	1	2	0	
TH01	6	170	212	44	118	
TH01	8	178	90	69	42	
TH01	8.3	181	0	0	1	
TH01	10	186	7	6	2	
TH01	R	NA	0	0	0	
vWA	13	163	0	5	2	
vWA	14	167	86	24	51	
...						

The first line needs to be present. The first column indicated the locus, the second the name of the allele, the third the fragment length. The next columns are the frequencies for specific ethnic groups. There are no limits to the number of ethnic groups included. If the name of an allele is `R`, then it is ignored.

If using the peak heights model, there should be an extra column, "LUS", which gives the mean longest uninterrupted sequence for that allele.

Value

A table, as read from the file.

See Also

read.unc.profile, read.known.profiles

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# get allele database
load.allele.database(kit=admin$kit)

## End(Not run)
```

`locus.likes` *likeLTD::locus.likes*

Description

Vector with individual locus likelihoods.

Usage

```
locus.likes(hypothesis,results,...)
```

Arguments

`hypothesis`	The hypothesis generated by either `prosecution.hypothesis` or `defence.hypothesis`.
`results`	Either prosecution or defence results from `evaluate` e.g. `results$Pros` or `results$Def`.
`...`	Any extra parameter that was passed to `evaluate` or `optimisation.params`.

Details

Convert the overall likelihood returned by DEoptim into locus specific likelihoods.

Value

Vector.

See Also

evaluate

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# Enter arguments
args = list(
        nUnknowns = 1,
        doDropin = FALSE,
        ethnic = "EA1",
        adj = 1,
        fst = 0.02,
        relatedness = c(0,0)
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis, append(admin,args))
hypD = do.call(defence.hypothesis, append(admin,args))

# Get parameters for optimisation
paramsP = optimisation.params(hypP)
paramsD = optimisation.params(hypD)

# Run optimisation
# n.steps set for demonstration purposes
results = evaluate(paramsP, paramsD, n.steps=1)

# get locus likelihoods under prosecution
locus.likes(hypP,results$Pros)

# get locus LRs
locus.likes(hypP,results$Pros)/locus.likes.peaks(hypD,results$Def)

## End(Not run)
```

---

locus.likes.peaks *likeLTD::locus.likes.peaks*

Description

Vector with individual locus likelihoods for peak height data.

Usage

```
locus.likes.peaks(hypothesis,results,...)
```

Arguments

hypothesis	The hypothesis generated by either prosecution.hypothesis.peaks or defence.hypothesis.peaks.
results	Either prosecution or defence results from evaluate.peaks e.g. results$Pros or results$Def.
...	Any extra parameter that was passed to evaluate.peaks.

Details

Convert the overall likelihood returned by evaluate.peaks into locus specific likelihoods.

Value

Vector.

See Also

evaluate.peaks

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# Enter arguments
args = list(
        nUnknowns = 1
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis.peaks, append(admin,args))
hypD = do.call(defence.hypothesis.peaks, append(admin,args))

# Get parameters for optimisation
paramsP = optimisation.params.peaks(hypP)
```

```
paramsD = optimisation.params.peaks(hypD)

# reduce number of iterations for demonstration purposes
paramsP$control$itermax=25
paramsD$control$itermax=25

# Run optimisation
# n.steps and converge set for demonstration purposes
results = evaluate.peaks(paramsP, paramsD, n.steps=1,
      converge=FALSE)

# get locus likelihoods under prosecution
locus.likes.peaks(hypP,results$Pros)

# get locus LRs
locus.likes.peaks(hypP,results$Pros)/locus.likes.peaks(hypD,results$Def)

## End(Not run)
```

`Objective Functions` *likeLTD objective functions*

Description

Objective functions giving the likelihoods computed by **likeLTD**. **These functions do not exist *per se*. They are created by** `create.likelihood` **and similar function.**

Details

The general workflow of **likeLTD** will generally result in the creation and maximization of one or more objective function. These functions will take the arguments listed above. They can also be accept any number of other arguments. This makes it easy both to extend **likeLTD** and to use objective functions with penalty functions.

Some of the arguments are verified for size and/or value.

The general form of these objective functions is:

```
function(locusAdjustment, power, dropout, degradation=NULL,              rcont=NULL, dropin=NULL, ...
```

- locusAdjustmentA scalar (single locus objective) or a list of values giving the locus adjustment to dropout for each locus. Must be positive.
- powerTvedebrink exponent. Must be a scalar and negative.
- dropoutDropout rate for each replicate. Must be between 0 and 1.
- degradationDegradation parameters to determine likeliness of dropout. Should be of length $k + u$.
- rcontRelative contribution. Must be of length $k + u$ or $k + u - 1$, where k is the number of contributors with known profiles, and u is the number of unknown contributors. Since the input is for *relative* contributions, there are two possible sizes. In the first case, the contribution

of a reference individual is kept to 1. In general, the reference individual will be the queried individual (if subjecto dropout and in prosecution hypothesis) or the first individual subject to dropout. In the second case, the contributions are used for all known and unknown contributors as given. It is for the user to know what she is doing. `rcont` should always be positive.

- dropinA scalar giving the dropin rate. Ignored in models without dropin. Otherwise it should be positive (unless given in exponential form).
- ... Any other parameter. Mostly ignored by objective functions, but it makes it easier to pass on parameters to the penalty function

---

`optimisation.params` *likeLTD::optimisation.params*

Description

Creates a list of parameters to use with `DEoptim::DEoptim`.

Usage

```
optimisation.params(hypothesis, verbose=FALSE, fixed=NULL,
                            logObjective=TRUE, logDegradation=TRUE,
                            arguments=NULL, zero=0, throwError=FALSE,
                      withPenalties=TRUE, doLinkage=TRUE,objective=NULL, iterMax=75,
                            likeMatrix=FALSE,...)
```

Arguments

`hypothesis`	Hypothesis from which to perform maximization
`verbose`	Wether to print likelihood each and every time the objective function is called
`fixed`	Names of the parameters to keep fixed
`logObjective`	If `TRUE` (default), the objective function is the log10-likelihood.
`logDegradation`	If `TRUE` (default), the degradation parameters are entered as 10^x
`arguments`	Initial parameters from which to start the maximization. If `NULL`, calls `initial.arguments`.
`zero`	Epsilon to indicate lower and upper bounds as $\alpha \pm \epsilon$ that exclude the bound itself
`throwError`	If TRUE, throws an error if the result is infinite
`withPenalties`	If TRUE, then penalties are evaluated and used
`doLinkage`	Logical indicating whether or not to apply a correction for linked loci. This correction is only applied when Q and X are assumed to be siblings i.e. `hypothesis$relatedness=c(0.5,0` This multiplies the prosecution likelihood by IMP_L/IMP_U, where IMP_L is the inverse match probability with linkage taken into account, and IMP_U is the same but without linkage taken into account.
`objective`	Objective function produced from create.likelihood.vectors
`iterMax`	Number of iterations to run the optimisation for
`likeMatrix`	Whether to return likelihoods for every genotype combination, or a likelihood summed over all genotypes after optimisation. Set to TRUE for individual genotype likelihoods. This is used for `get.likely.genotypes`.
`...`	Any named parameter to modify the hypothesis, e.g. `nUnknowns`

Details

Starting from the hypothesis, it creates an list of arguments which can be applied to `DEoptim::DEoptim` to obtain the maximum (log-)likelihood of that hypothesis.

It accepts a number of customization:

- The optimisation can be performed for the likelihood or the log of the likelihood. The latter is recommended.
- wether the `degradation` parameter should be inputs as x or as an exponent 10^x. The latter seems to be more numerically stable, likely because degradations (in first form) are factors of an exponent in any case.
- whether to keep some nuisance parameters fixed

In any case, the value returned can always be modified prior to calling `DEoptim::DEoptim`.

Value

`fn`	The objective function
`lower`	Lower bounds for the parameters
`upper`	Upper bounds for the parameters
`control`	Control parameters for `DEoptim::DEoptim`

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# Enter arguments
args = list(
        nUnknowns = 1,
        doDropin = FALSE,
        ethnic = "EA1",
        adj = 1,
        fst = 0.02,
        relatedness = c(0,0)
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis, append(admin,args))
hypD = do.call(defence.hypothesis, append(admin,args))
```

```
# Get parameters for optimisation
paramsP = optimisation.params(hypP)
paramsD = optimisation.params(hypD)

## End(Not run)
```

---

optimisation.params.peaks

likeLTD::optimisation.params.peaks

Description

Creates a list of parameters to use with `DEoptim::DEoptim` for peak height data.

Usage

```
optimisation.params.peaks(hypothesis, verbose=TRUE, fixed=NULL,
                          logObjective=TRUE, logDegradation=TRUE,
                          arguments=NULL, zero=1e-6, throwError=FALSE,
                     withPenalties=TRUE, doLinkage=TRUE,objective=NULL, iterMax=25,
                          likeMatrix=FALSE,diagnose=FALSE,DEoptimStrategy=3,
   searchPopFactor=1,DEoptimF=0.8,DEoptimC=NULL,
maxDropin=100,...)
```

Arguments

`hypothesis`	Hypothesis from which to perform maximization
`verbose`	Wether to print likelihood each and every time the objective function is called
`fixed`	Names of the parameters to keep fixed
`logObjective`	If `TRUE` (default), the objective function is the log10-likelihood.
`logDegradation`	If `TRUE` (default), the degradation parameters are entered as 10^x
`arguments`	Initial parameters from which to start the maximization. If `NULL`, calls `initial.arguments`.
`zero`	Epsilon to indicate lower and upper bounds as $\alpha \pm \epsilon$ that exclude the bound itself
`throwError`	If TRUE, throws an error if the result is infinite
`withPenalties`	If TRUE, then penalties are evaluated and used
`doLinkage`	Logical indicating whether or not to apply a correction for linked loci. This correction is only applied when Q and X are assumed to be siblings i.e. `hypothesis$relatedness=c(0.5,0` This multiplies the prosecution likelihood by IMP_L/IMP_U, where IMP_L is the inverse match probability with linkage taken into account, and IMP_U is the same but without linkage taken into account.
`objective`	Objective function produced from create.likelihood.vectors.peaks
`iterMax`	Number of iterations to run the optimisation for before checking for convergence within each step.

likeMatrix	Whether to return likelihoods for every genotype combination, or a likelihood summed over all genotypes after optimisation. Set to TRUE for individual genotype likelihoods. This is used for get.likely.genotypes.peaks.
diagnose	Logical. If TRUE a list of peak heights, estimated mean peak heights and standard deviation of peak heights is returned by result$fn.
...	Any named parameter to modify the hypothesis, e.g. nUnknowns
DEoptimStrategy	Parameter to control the strategy used by DEoptim. See DEoptim::DEoptim.control.
searchPopFactor	The population size used by DEoptim is searchPopFactor multiplied by the number of parameters to optimise over.
DEoptimF	Parameter to control differential weighting of DEoptim. See DEoptim::DEoptim.control.
DEoptimC	Parameter to control the speed of crossover adaptation for DEoptim. See DEoptim::DEoptim.control.
maxDropin	Upper bound for the dropin parameter to be used by evaluate.peaks. If you wish to explain a minor cnotributor as dropin this value may need to be greater than the default of 100.

Details

Starting from the hypothesis, it creates a list of arguments which can be applied to DEoptim::DEoptim to obtain the maximum (log-)likelihood of that hypothesis.

It accepts a number of customizations:

- The optimisation can be performed for the likelihood or the log of the likelihood. The latter is recommended.
- whether the degradation parameter should be inputs as x or as an exponent 10^x. The latter seems to be more numerically stable, likely because degradations (in first form) are factors of an exponent in any case.
- whether to keep some nuisance parameters fixed.

In any case, the value returned can always be modified prior to calling evaluate.peaks.

Value

fn	The objective function
lower	Lower bounds for the parameters
upper	Upper bounds for the parameters
control	Control parameters for DEoptim::DEoptim

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# Enter arguments
args = list(
        nUnknowns = 1
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis.peaks, append(admin,args))
hypD = do.call(defence.hypothesis.peaks, append(admin,args))

# Get parameters for optimisation
paramsP = optimisation.params.peaks(hypP)
paramsD = optimisation.params.peaks(hypD)

# parameters without linkage adjustment
paramsP = optimisation.params.peaks(hypP,doLinkage=FALSE)

## End(Not run)
```

`output.report` *likeLTD::output.report*

Description

Outputs a docx report of results.

Usage

```
output.report(prosecutionHypothesis,defenceHypothesis,
                      results,file=NULL)
```

Arguments

`prosecutionHypothesis`	The output from `prosecution.hypothesis`.
`defenceHypothesis`	The output from `defence.hypothesis`.
`results`	The output from `evaluate`.
`file`	A file name for the output report. By default a filename is created to avoid over writing a previous file.

Details

Creates a docx containing all results, placed in the directory specified by admin$outputPath. Results include both the pre-evaluation assessment of the alleles from both the reference profiles and the crime scene profile, and the post-evaluation results of the likelihoods and weight of evidence.

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# Enter arguments
args = list(
        nUnknowns = 1,
        doDropin = FALSE,
        ethnic = "EA1",
        adj = 1,
        fst = 0.02,
        relatedness = c(0,0)
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis, append(admin,args))
hypD = do.call(defence.hypothesis, append(admin,args))

# Get parameters for optimisation
paramsP = optimisation.params(hypP)
paramsD = optimisation.params(hypD)

# Run optimisation
# n.steps set for demonstration purposes
results = evaluate(paramsP, paramsD, n.steps=1)

# Generate output report
output.report(hypP,hypD,results)

## End(Not run)
```

---

output.report.peaks *likeLTD::output.report.peaks*

Description

Outputs a docx report of results.

Usage

```
output.report.peaks(prosecutionHypothesis,defenceHypothesis,
                      results,file=NULL,figRes=300)
```

Arguments

prosecutionHypothesis	The output from prosecution.hypothesis.peaks.
defenceHypothesis	The output from defence.hypothesis.peaks.
results	The output from evaluate.peaks.
file	A file name for the output report. By default a filename is created to avoid over writing a previous file.
figRes	Resolution of figures in the output report. Defaults to 300 DPI.

Details

Creates a docx containing all results, placed in the directory specified by admin$outputPath. Results include both the pre-evaluation assessment of the crime-stain and reference profiles, and the post-evaluation results of the likelihoods and weight of evidence.

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# Enter arguments
args = list(
        nUnknowns = 1
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis.peaks, append(admin,args))
hypD = do.call(defence.hypothesis.peaks, append(admin,args))

# Get parameters for optimisation
paramsP = optimisation.params.peaks(hypP)
```

```
paramsD = optimisation.params.peaks(hypD)

# reduce number of iterations for demonstration purposes
paramsP$control$itermax=25
paramsD$control$itermax=25

# Run optimisation
# n.steps and converge set for demonstration purposes
results = evaluate.peaks(paramsP, paramsD, n.steps=1,
      converge=FALSE)

# generate output report
output.report.peaks(hypP,hypD,results)

## End(Not run)
```

`pack.admin.input` *likeLTD::pack.admin.input*

Description

Packs and verifies administrative information. Only used in allele report.

Usage

```
pack.admin.input(cspFile, refFile, caseName='dummy',
                         databaseFile=NULL, kit=NULL, linkageFile=NULL,
                          outputPath=getwd())
```

Arguments

`cspFile`	Mixed crime scene profile.
`refFile`	Reference profiles.
`caseName`	Name of the current case.
`databaseFile`	Path to the allele database.
`kit`	Database to use if `databaseFile` not specified.
`linkageFile`	Path to file with recombination rates for linked loci.
`outputPath`	Path where the output should be stored.

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
```

```
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

## End(Not run)
```

`pack.admin.input.peaks` *likeLTD::pack.admin.input.peaks*

Description

Packs and verifies administrative information. Can be used as partial input for `defence.hypothesis.peaks` and `prosecution.hypothesis.peaks`

Usage

```
pack.admin.input.peaks(peaksFile, refFile, caseName='dummy',
                       databaseFile=NULL, kit=NULL, linkageFile=NULL,
                       detectionThresh=20,outputPath=getwd())
```

Arguments

`peaksFile`	Crime scene profile with peak height data
`refFile`	Reference profiles
`caseName`	Name of the current case
`databaseFile`	Path to the allele database
`kit`	Database to use if `databaseFile` not specified
`linkageFile`	Path to file with recombination rates for linked loci
`outputPath`	Path where the output should be stored. Defaults to current working directory.
`detectionThresh`	The detection threshold used to analyse peak heights from electrophoresis. This may be a single value, or a list of one value per locus. No peaks in the CSP should have a peak height below this value if a single value. No peaks in a given locus should have a peak height below the specified locus value.

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
              detectionThresh = 20
               )

## End(Not run)
```

`peaks.results.plot` *likeLTD::peaks.results.plot*

Description

Plot distribution of fitted peaks along with actual peak heights.

Usage

```
peaks.results.plot(hyp,res,replicate=1,toplot=NULL,fileName=NULL,...)
```

Arguments

`hyp`	Hypothesis used to generate parameters for optimisation.
`res`	Either prosecution or defence results from `evaluate.peaks` e.g. `results$Pros` or `results$Def`.
`replicate`	Which replicate results to plot.
`toplot`	Integer vector indicating which loci to plot. If `NULL`, all loci are plotted.
`fileName`	Output file name for plot. If `NULL` the plot is output to a plot window instead.
`...`	Extra parameters to pass to `boxplot`.

Details

CSP peak heights for a single replicate are plotted, with boxplots representing the distribution for each peak height estimated through optimisation.

Value

Pdf file or plot window.

See Also

plot.CSP.heights

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# Enter arguments
args = list(
        nUnknowns = 1
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis.peaks, append(admin,args))
hypD = do.call(defence.hypothesis.peaks, append(admin,args))

# Get parameters for optimisation
paramsP = optimisation.params.peaks(hypP)
paramsD = optimisation.params.peaks(hypD)

# reduce number of iterations for demonstration purposes
paramsP$control$itermax=25
paramsD$control$itermax=25

# Run optimisation
# n.steps and converge set for demonstration purposes
results = evaluate.peaks(paramsP, paramsD, n.steps=1,
      converge=FALSE)

# plot fitted results under prosecution
peaks.results.plot(hypP,results$Pros)

# plot fitted results under defence
peaks.results.plot(hypD,results$Def)

# plot fitted results for first four loci
peaks.results.plot(hypP,results$Pros,toplot=1:4)

## End(Not run)
```

---

penalties *likeLTD::penalties*

Description

Returns the penalty for the current arguments

Usage

```
penalties(locusAdjustment, power, dropout, degradation=NULL,
                rcont=NULL, dropin, locusAdjPenalty=50, dropinPenalty=2,
                degradationPenalty=50, bemn=-4.35, besd=0.38, ...)
```

Arguments

`locusAdjustment`	Locus adjustment for each locus
`power`	Tvedebrink exponent
`dropout`	Ignored
`degradation`	Degradation parameters
`rcont`	Ignored.
`dropin`	Dropin rate
`locusAdjPenalty`	Penalty parameter for the locus adjustments
`dropinPenalty`	Penalty parameter for the dropin rate
`degradationPenalty`	Penalty parameter for the degradation parameters
`bemn`	Mean of the normal distribution used to penalize degradation
`besd`	Standard deviation of the normal distribution used to penalize degradation
`...`	Ignored

Details

The penalties are applied if and only if the relevant arguments (locusAdjustment, dropin, degradation, power) are provided. The penalties are as follows:

- dropin:e^{-d*p} where d is the dropin rate and p the associated penalty. The values is normalized to one locus.
- degradation:$e^{-p\sum d}$ where d are the degradation values and p is the associated penalty
- power:`dnorm(t, bemn, besd)` where `t` is the Tvedebrink exponent, `dnorm` is the density of the normal distribution with mean `bemn` and standard deviation `besd`
- locusAdjustment:`dgamma(a, p, p)` where `a` is the locus adjustment, `dgamma` is the density of the Gamma distribution with `p` its shape and rate

Some of these penalties are meant to be applied simultaneously across all loci. Since we want penalties *per* locus, a normalization $p^{\frac{1}{n}}$ is applied, where p is the penalty and n the number of loci.

Value

An array of penalties per locus

See Also

create.likelihood.vectors, create.likelihood.log, create.likelihood, Objective Functions

---

`plotLikelihood.2d` *likeLTD::plotLikelihood.2d*

Description

Plots two dimensional graph of likelihood for two input directions.

Usage

```
plotLikelihood.2d(hypothesis, which=c(1, 2), large=100, N=20,
                          arguments=NULL, x=NULL, y=NULL, logObjective=TRUE,
                          logDegradation=TRUE, contours=list(), ...)
```

Arguments

`hypothesis`	Hypothesis from which to derive the likelihood
`which`	Indices of the arguments for which to plot likelihood
`large`	Number with which to replace Inf in upper and lower bounds. Can be a 2-tuple.
`N`	Number of points per axis. Can be a 2-tuple.
`arguments`	Arguments to the likelihood. Defines values for dimensions that are not plotted. Defaults to `initial.arguments(...)`.
`x`	Range over which to plot abcissa
`y`	Range over which to plot ordinate
`logObjective`	Whether to plot log-likelihood
`logDegradation`	Whether to input degradation as 10^x
`contours`	Arguments to stat_contour
`...`	Other arguments are fed to the internal call to optimisation.params

`prosecution.hypothesis` *likeLTD::prosecution.hypothesis*

Description

Helper function to create the input for the prosecution.

Usage

```
prosecution.hypothesis(cspFile, refFile, ethnic='NDU1', nUnknowns=0,
                       adj=1e0, fst=0.02, databaseFile=NULL,
                       linkageFile=NULL,
                       doDropin=FALSE,
                       combineRare=TRUE, rareThreshold=0.05,
                       kit=NULL, relationship=0, ...)
```

Arguments

`cspFile`	Path to the crime scene profile.
`refFile`	Path to the known profiles.
`ethnic`	Ethnicity, e.g. subgroup within the allele frequency database.
`nUnknowns`	Number of unknown contributors for which to perform calculation.
`adj`	Allele frequency adjustment parameter.
`fst`	Allele fraction adjustment F_{ST}.
`databaseFile`	Path to the allele database. If NULL, then defaults to the NGMSelect database provided with `likeLTD`.
`linkageFile`	Path to recombination rate information. If NULL then defaults to the linkage file provided with `likeLTD`. Only used if `relatedness=c(0.5,0.25)` i.e. when Q and X are siblings.
`relationship`	Specified relationship between Q and X. Can take values of 0=unrelated, 1=parent/offspring, 2=siblings, 3=uncle/nephew, 4=half-uncle/half-nephew, 5=cousins, 6=grandparent/grandchild, 7=half-siblings.
`doDropin`	Whether or not to model drop-in.
`combineRare`	Whether or not to combine rare unobserved alleles into a single allele.
`rareThreshold`	If `combineRare=TRUE`, this is the probability threshold below which an allele is classed as rare, and therefore combined with other rare unobserved alleles.
`kit`	Parameter specifying which allele database supplied with likeLTD to use if `linkageFile` is not specified. Possibilities are "DNA17", "Identifiler" and "SGMplus".
`...`	Other parameters to determine how to perform calculations.

Details

It loads the CSP, known profiles, and allele database from file. It makes sure the relatedness factor is zero. It sets the reference individual to the first queried individual, unless it is not subject to dropout. In that case, the reference individual is the first individual with dropout.

Value

A list of named input parameters, or hypothesis, suitable for the prosecution.

See Also

defence.hypothesis

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# Enter arguments
args = list(
        nUnknowns = 1,
        doDropin = FALSE,
        ethnic = "EA1",
        adj = 1,
        fst = 0.02,
        relatedness = c(0,0)
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis, append(admin,args))
hypD = do.call(defence.hypothesis, append(admin,args))

## End(Not run)
```

---

prosecution.hypothesis.peaks

likeLTD::prosecution.hypothesis.peaks

Description

Helper function to create the input for the prosecution using peak height data.

Usage

```
prosecution.hypothesis.peaks(peaksFile, refFile, ethnic='NDU1', nUnknowns=0,
                        adj=1e0, fst=0.03,
                        linkageFile=NULL, databaseFile=NULL,
                        detectionThresh=20, doDropin=FALSE,
                        doDoubleStutter=TRUE,doOverStutter=TRUE,
                        combineRare=TRUE, rareThreshold=1,
                        kit=NULL, relationship=0, ...)
```

Arguments

peaksFile	Path to the crime scene profile.
refFile	Path to the known profiles.
ethnic	Ethnicity, e.g. subgroup within the allele frequency database.
nUnknowns	Number of unknown contributors for which to perform calculation.
adj	Allele frequency adjustment parameter.
fst	Allele fraction adjustment F_{ST}.
databaseFile	Path to the allele database. If NULL, then defaults to the NGMSelect database provided with likeLTD. Must include longest uninterrupted sequence (LUS) values for alleles.
linkageFile	Path to recombination rate information. If NULL then defaults to the linkage file provided with likeLTD. Only used if relationship is not 0 or 1 i.e. when Q and X are closely related but not parent/offspring.
relationship	Specified relationship between Q and X. Can take values of 0=unrelated, 1=parent/offspring, 2=siblings, 3=uncle/nephew, 4=half-uncle/half-nephew, 5=cousins, 6=grandparent/grandchild, 7=half-siblings.
detectionThresh	Detection threshold for peaks. Can be a single value, or a named list containing one value per locus.
doDropin	Whether or not to model drop-in. Note dropin is not currently possible with the peak heights model.
doDoubleStutter	Logical. Whether or not to model double stutter.
doOverStutter	Logical. Whether or not to model over stutter.
combineRare	Whether or not to combine rare unobserved alleles into a single allele.
rareThreshold	If combineRare=TRUE, this is the probability threshold below which an allele is classed as rare, and therefore combined with other rare unobserved alleles.
kit	Parameter specifying which allele database supplied with likeLTD to use if linkageFile is not specified. Possibilities are "DNA17-lus".
...	Other parameters to determine how to perform calculations.

Details

It loads the CSP, known profiles, and allele database from file.

Value

A list of named input parameters, or hypothesis, suitable for the prosecution.

See Also

defence.hypothesis.peaks

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# Enter arguments
args = list(
        nUnknowns = 1
        )

# Create hypotheses
hypP = do.call(prosecution.hypothesis.peaks, append(admin,args))
hypD = do.call(defence.hypothesis.peaks, append(admin,args))

## End(Not run)
```

`read.csp.profile` *likeLTD::read.csp.profile*

Description

Reads the Crime Scene Profile from a CSV file.

Usage

```
read.csp.profile(path)
```

Arguments

`path`	Path to the crime scene profile.

Details

The input is a file in the CSV format (comma-separated values). It should have the following form:

Stain	Profiling system	Plate/Run	Allelic/Uncertain	Locus	Locus	...
IGNORED	IGNORED	IGNORED	Allelic	"13,14,15"	"16,16"	...
IGNORED	IGNORED	IGNORED	Uncertain		"17.2"	...

Columns are separted by commas (not present in the table above). The first line, containing the name of the columns need be present. The first three columns are ignored for the purpose of the **likeLTD** package. They are present (and expected) merely for convenience. The fourth column indicates whether a line contains certain or uncertain alleles. "Uncertain" implies the latter, anything else implies the former. Beware of the capitalization. The next columns are for loci. "Locus" should be replaced with the actual locus name in the first line. Other lines should contain the comma-separated names of the alleles present in the CSP, within quotation marks. Entries can be empty. The lines labeled "Uncertain" are ignored by this function.

Value

A matrix where columns are loci and rows are replicates. Each element is a vector of character, and each character is the name of an allele *certainly* present in the CSP for that loci and replicate. For simplicity, loci for which all entries are NA are removed.

See Also

read.unc.profile, read.known.profiles

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# get CSP
read.csp.profile(admin$cspFile)

## End(Not run)
```

`read.known.profiles` *likeLTD::read.known.profiles*

Description

Reads the known profiles from file.

Usage

```
read.known.profiles(path)
```

Arguments

path	Path to a CSV file with known profiles.

Details

The input is a file in the CSV format (comma-separated values). It should have the following form:

Individual	known/queried		Locus	Locus	...
name	queried	"13,15"	"16,16"	...	
name	known		"18, 19"	"16, 17.2"	...

Columns are separted by commas (not present in the table above). The first line, containing the name of the columns need be present. Each row is the profile of a different individual. Names or identifiers for the individuals are contained in the first column. The second column should contain the word "queried" if the individual should be queried. The other columns are for the differnt loci. "Locus" should be replaced with the name of the locus in the first column. Each item consists of two comma separated values within qutoation marks. The values are the name of the allles in that individual's profile for that locus.

Value

A matrix where columns are loci and rows are individual profiles. Each element is a vector of two characters, and each character is the name of an allele represented in the individual. There should be two characters for each entry.

Additionnaly, the first column (named "queried") indicates whether that individiual is to be queried (TRUE) or is known to be in the CSP (FALSE).

See Also

read.csp.profile, read.unc.profiles

Examples

```
## Not run:
######################
# Peak heights model #
######################
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# get reference profiles
read.known.profiles(admin$refFile)

##################
# Discrete model #
##################
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# get reference profiles
read.known.profiles(admin$refFile)

## End(Not run)
```

`read.peaks.profile` *likeLTD::read.peaks.profile*

Description

Reads the Crime Scene Profile from a CSV file of peak height data.

Usage

```
read.peaks.profile(FILE)
```

Arguments

`FILE`	Path to the crime scene profile with peak height data.

Details

The input is a file in the CSV format (comma-separated values). It should have the following form:

Sample File	Marker	Allele 1	...	Allele x	Height 1	...	Height x
sample 1	locus 1	9.3	...	11	151	...	190
...	...	...	...	...	...	...	...
sample 1	locus n	23	...	NA	301	...	NA

sample r	locus 1	9.3	...	11	132	...	123
...	...	...	...	...	...	...	...
sample r	locus n	23	...	NA	256	...	NA

Columns are separted by commas (not present in the table above). The first line, containing the names of the columns needs to be present. x, r and n can be any number. Columns headed by "Allele 1 ... x" should contain the allelic designation of the peak. Columns headed by "Height 1 ... x" should contain the peak height in RFU.

Value

A list of three elements, `alleles`, `heights` and `sizes`. Each element is itself a list with r elements, where r is the number of replicates in the CSP. Each of these elements is a matrix of n by x where n is the number of loci in the CSP, and x is the maximum number of peaks recorded at any locus for that replicate.

See Also

read.known.profiles

Examples

```
## Not run:
# datapath to example files
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 = "Laboratory",
            detectionThresh = 20
             )

# read csp
csp = read.peaks.profile(admin$peaksFile)

## End(Not run)
```

`read.unc.profile` *likeLTD::read.unc.profile*

Description

Reads the uncertain alleles from the Crime Scene Profile from file.

Usage

```
read.unc.profile(path)
```

Arguments

path	Path to the crime scene profile.

Details

The input is a CSV file. It is (or can be) the same file as for `read.csv.profile`. In this case, the lines labeled "Uncertain" are selected.

Value

A matrix where columns are loci and rows are replicates. Each element is a vector of character, and each character is the name of an allele *which seems to be present but at a level too low for certainty* in the CSP for that loci and replicate.

See Also

read.csp.profile, read.known.profiles

Examples

```
## Not run:
# datapath to example files
datapath = file.path(system.file("extdata", package="likeLTD"),"hammer")

# File paths and case name for allele report
admin = pack.admin.input(
            cspFile = file.path(datapath, 'hammer-CSP.csv'),
            refFile = file.path(datapath, 'hammer-reference.csv'),
            caseName = "hammer",
    kit= "SGMplus"
             )

# get uncertain profile
read.unc.profile(admin$cspFile)

## End(Not run)
```

---

relistArguments *likeLTD::relistArguments*

Description

In practice, this function undoes the flattening needed to run optim. Hence, it should take the same parameters that optimisation.params does. Takes a linear vector of parameters, as passed to and returned from an optimisation method, and makes it a list. Transforms degradation parameters back into normal (non-logarithmic) form. Adds fixed arguments back into the list.

Usage

```
relistArguments(parameters, hypothesis, fixed=NULL,
logDegradation=TRUE, arguments=NULL)
```

Arguments

parameters	Vector of parameters
hypothesis	Hypothesis from which objective function was obtained.
fixed	Names of the arguments which were fixed during optimisation.
logDegradation	Whether degradation is logarithmic form.
arguments	Initial guess, if any, when starting minimization.

Value

Input parameters as a list.

See Also

optimisation.params, initial.arguments

---

relistArguments.peaks *likeLTD::relistArguments.peaks*

Description

In practice, this function undoes the flattening needed to run DEoptim. Hence, it should take the same parameters that optimisation.params.peaks does. Takes a linear vector of parameters, as passed to and returned from an optimisation method, and makes it a list. Transforms degradation parameters back into normal (non-logarithmic) form. Adds fixed arguments back into the list.

Usage

```
relistArguments.peaks(parameters, hypothesis, fixed=NULL,
logDegradation=TRUE, arguments=NULL)
```

Arguments

parameters	Vector of parameters
hypothesis	Hypothesis from which objective function was obtained.
fixed	Names of the arguments which were fixed during optimisation.
logDegradation	Whether degradation is logarithmic form.
arguments	Initial guess, if any, when starting minimization.

Value

Input parameters as a list.

See Also

optimisation.params.peaks, initial.arguments.peaks

`SGMplus-db` *SGMplus allele database*

Description

A table with allele frequencies for SGMplus loci. Populations are "EA1" (Caucasian), "EA3" (African/Afro-Caribbean), "EA4" (South Asian).

Format

A table, as described in `load.allele.database`.

`unitTests.likeLTD` *Unit tests for the package likeLTD*

Description

Performs unit tests defined in this package by running `example(unitTests.likeLTD)`. Tests are in `runit*.R` files Located in the '/unitTests' subdirectory or one of its subdirectories ('/inst/unitTests' and subdirectories in package sources).

Author(s)

John Doe (<john.doe@doelabs.org>)

Examples

```
## Not run:
if (require(svUnit)) {
clearLog()
runTest(svSuite("package:likeLTD"))
## Possibly run other tests here...
errorLog()
}

## End(Not run)
```

Index