Package 'pcrsim'

October 3, 2016

Type Package

Title Simulation of the Forensic DNA Process

Version 1.0.1

Date 2016-10-03

Author Oskar Hansson

Maintainer Oskar Hansson <oskar.hansson@fhi.no>

URL https://github.com/OskarHansson/pcrsim

BugReports https://github.com/OskarHansson/pcrsim/issues

Imports ggplot2, data.table, gWidgets, mc2d, plyr

Depends strvalidator (>= 1.6)

Description Simulate the forensic DNA process: generate random or fixed DNA profiles, create forensic samples including mixtures of diploid and haploid cells, simulate DNA extraction, normalization, degradation, amplification including stutters and inter-locus balance, and capillary electrophoresis. DNA profiles are visualized as electropherograms and saved in tables. The command pcrsim() opens up a graphical user interface which allow the user to create projects, to enter, load, and save parameters required for the simulation. The simulation is transparent and the parameters used in each step of the simulation can be viewed in the result tables.

License GPL-2

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-10-03 13:13:20

R topics documented:

`pcrsim-package` *Simulation of the Forensic DNA process*

Description

Forensic DNA process simulator.

Details

PCRsim is a package to simulate the forensic DNA process. The function `pcrsim` opens up a graphical user interface which allow the user to enter parameters required for the simulation. Once calibrated the program can potentially be used to: reduce the laboratory work needed to validate new STR kits, create samples for educational purposes, help develop methods for interpretation of DNA evidence, etc.

This is a first version which is still experimental and under development.

Areas in need of more research are better calibration and more correct scaling to peak heights over a range of input amounts. The current implementation is built to mimic the biological processes as closely as possible and are not suitable for simulation of large number of samples due to performance.

Author(s)

Oskar Hansson <oskar.hansson@fhi.no>

References

Gill, Peter, James Curran, and Keith Elliot. \u0022 A Graphical Simulation Model of the Entire DNA Process Associated with the Analysis of Short Tandem Repeat Loci\u0022 Nucleic Acids Research 33, no. 2 (2005): 632-643. doi:10.1093/nar/gki205.

calculateDegradation *Calculate Degradation Parameter*

Description

Calculate the degradation parameter (probability of degradation per base pair).

Usage

```
calculateDegradation(conc, size, debug = FALSE)
```

Arguments

conc	numeric vector with measured concentrations.
size	numeric vector with number of base pairs for the targets.
debug	logical to print debug information.

Details

Calculates the degradation parameter given the concentrations measured with two targets of different size. NB! The concentration from the shorter fragment must be given first with the corresponding target size first in the size vector.

Value

numeric calculated degradation parameter.

Examples

```
# The DNA concentration for a degraded sample measured with probe sizes 70 and 220 bp
# was 85 and 0.5 ng/ul respectively.
# Calulate the degradation parameter:
calculateDegradation(conc=c(85,0.5), size=c(70,220))
```

calibrateLb *Calibrate Inter-locus Balance*

Description

Estimate values for the PCR efficiency parameter per locus that satisfy the target inter-locus balances.

Usage

```
calibrateLb(sim = 100, target, amount = 0.5, cell.dna = 0.006,
  pcr.cyc = 30, acc.dev = 0.001, step.size = 0.001, seed = 0.85,
  max.eff = 0.98, progress = TRUE, debug = FALSE)
```

Arguments

sim	integer the number of simulations per calibration cycle.
target	numeric vector with target interlocus balances.
amount	numeric for amount of DNA in ng.
cell.dna	numeric the DNA content of a diploid cell in nanograms (default is 0.006 ng).
pcr.cyc	integer the number of PCR cycles.
acc.dev	numeric, accepted deviation from target.
step.size	numeric, the probability of PCR is changed by this value.
seed	numeric, start value for optimisation of the PCR probability.
max.eff	numeric, maximal value for estimated PCR efficiency.
progress	logical, print progress to console.
debug	logical to print debug information.

Details

The inter-locus balance for a kit should be characterised during the internal validation of the kit. The function search for PCR efficiency values per locus that upon simulation are similar to the target inter-locus balances. Use the PCR efficiency value obtained from the `calibratePCRsim` function as seed value.

Value

vector with estimated PCR efficiencies for each locus.

Examples

```
# Experimental inter-locus balances for the STR kit to be simulated (sums to 1).
target <- c(0.20, 0.10, 0.15, 0.25, 0.30)

# Find PCR efficiency values that upon simulation
# satisfy the experimental data for 0.5 ng of input DNA.
set.seed(10) # For reproducibility.
calibrateLb(sim=10, target=target, amount=0.5, seed=0.85, progress=FALSE)

# Locus specific PCR efficency parameters can now be used as parameters.
# [1] 0.858 0.816 0.841 0.871 0.883
```

`calibratePCRsim` *Calibrate PCRsim*

Description

Estimates the detection threshold, peak height scaling factor, and PCR efficiency needed to calibrate PCRsim.

Usage

```
calibratePCRsim(data, target = NULL, ref = NULL, quant = NULL,
  ignore.case = TRUE, kit = NULL, db = NULL, fixed.profile = NULL,
  sim = 1, pcr.cyc, ce.aliq = 1, pcr.aliq = 17.5, pcr.vol = 25,
  minimize = TRUE, step.size = 0.001, cell.dna = 0.006,
  dna.amount = 0.5, filter = TRUE, plot.data = TRUE, decimals = 4,
  debug = FALSE, ext.debug = FALSE)
```

Arguments

`data`	data.frame with observed DNA result. Required columns are sample names ('Sample.Name'), average peak height ('H'), and (mean) DNA concentration ('Mean').
`target`	integer average peak height ('H') as observed average across replicate analyses of samples with `dna.amount` of DNA. If NULL it will be estimated from the linear regression at `dna.amount` of DNA.
`ref`	data.frame with known profiles for the samples in `data`. Required columns are 'Sample.Name', 'Marker', and ('Allele'), and (mean) DNA concentration ('Mean').
`quant`	data.frame with (average) 'Concentration' or 'Amount' for the samples in `data`.
`ignore.case`	logical `TRUE` to ignore case in sample name matching.
`kit`	character string to specify the STR DNA typing kit to simulate. If NULL all markers in `db` will be used.
`db`	data.frame with the allele frequency database (if random profiles are simulated).
`fixed.profile`	data.frame with columns 'Marker' and 'Allele' (if fixed profiles are simulated).
`sim`	integer the number of simulations per calibration cycle.
`pcr.cyc`	integer the number of PCR cycles.
`ce.aliq`	integer the aliquot PCR product used for capillary electrophoresis.
`pcr.aliq`	integer the aliquot DNA extract transferred to the PCR reaction.
`pcr.vol`	integer the total PCR reaction volume.
`minimize`	logical `TRUE` stops when the squared difference is minimized, `FALSE` continues until the PCR efficiency is 0.
`step.size`	numeric size of PCR efficiency reduction for each calibration cycle.

cell.dna	numeric the DNA content of a diploid cell in nanograms (default is 0.006 ng).
dna.amount	numeric the amount of DNA in nanograms (ng) to be used in simulation. NB! must correspond to the amount in samples used to calculate target.
filter	logical TRUE to retrieve known alleles defined in ref from data.
plot.data	logical to show linear regression data plot with marked target.
decimals	interger for number of decimal places in plot title.
debug	logical to print debug information.
ext.debug	logical to print extended debug information.

Details

To calibrate the PCR simulator perform the following experiments: 1) Prepare single source samples with optimal amount of DNA (different or replicates) Calculate the average total peak height (sum of peak heights) across all samples to set the target parameter. 2) Prepare a serial dilution (preferrably from intact cells, or from single source crime scene samples). It is suitable to go from approximately optimal amount down to low concentrations with drop-outs or completely blank profiles. Quantify each dilution as accurately as possible. Amplify using normal procedure and analyse the PCR product. Require a dataset with sample names ('Sample.Name'), and (average) concentration or (average) amount in columns named 'Concentration' and 'Amount' respectively. Also requires the average peak height ('H').

NB! Samples with either zero average peak height or zero number of molecules is removed automatically. In addition, if replicate quants, samples where one replicate was negative can be removed manually.

---

calibrateScaling *Calibrate Peak Height Scaling*

Description

Corrects the peak height scaling intercept.

Usage

```
calibrateScaling(data, ref = NULL, target, c.min = 0, c.max = 10000,
  min.step.size = 1e-04, exact.matching = FALSE, progress = FALSE,
  .i = 1)
```

Arguments

data	data.frame simulated data.
ref	data.frame with reference profiles for the simulated data. If NULL the best guess will be used (see guessProfile).
target	numeric target average peak height 'H'.
c.min	numeric the smallest correction factor in the current range.

c.max	numeric the largest correction factor in the current range.
min.step.size	numeric threshold. Exit function when step size is this small.
exact.matching	logical to indicate exact reference sample to dataset sample name matching.
progress	logical flag to show progress messages.
.i	integer internal counter for number of iterations.

Details

Finds the scaling factor correction that gives the best fit between the simulated peak heights and the target average peak height. This is the final step in calibrating pcrsim for a specific kit and method.

Value

numeric corrected scaling intercept.

---

compact *Compact*

Description

Add Identical Peaks.

Usage

```
compact(data, per.sample = TRUE, col = "Height", sim = FALSE,
  debug = FALSE)
```

Arguments

data	data frame with at least 'Sample.Name', 'Marker', 'Allele', and 'Height' columns.
per.sample	logical TRUE compact per sample, FALSE for entire dataset (in which case 'Sample.Name' is set to 'Profile' and if sim=TRUE 'Sim' is set to '1' and 'PCR.Vol' is set to the mean).
col	character string to specify which column to compact.
sim	logical to specify if simulation columns should be replicated (i.e. 'Sim' and 'PCR.Vol').
debug	logical for printing debug information.

Details

Add peak heights of identical allele designations together.

Value

data.frame with columns 'Sample.Name', 'Marker', 'Allele', and 'Height'.

compactStutter *Merge Stutters With Profile.*

Description

Add peak heights of stutters to the peak heights of alleles.

Usage

```
compactStutter(data, targetcol = NA, stutterin = "PCR.Stutter.",
  debug = FALSE)
```

Arguments

data	data.frame to be merged. Columns 'Sample.Name', 'Marker', and 'Allele' are required.
targetcol	character name for target column.
stutterin	character name for stutter column.
debug	logical for printing debug information.

Details

Merge stutter values to alleles in a simulated dataset.

Value

data.frame with simulation results in columns 'CE.xxx'.

getParameter *Get Kit Parameters*

Description

Provides parameters for simulation for different STR kits and methods.

Usage

```
getParameter(kit = NULL, what = NA, method = NA, show.messages = FALSE,
  .kit.param = NULL, debug = FALSE)
```

Arguments

kit	string or integer specifying the kit.
what	string to specify which information to return. Default is 'NA' which return all info. Not case sensitive.
method	string to specify which method to return. Default is 'NA' which return all info. Not case sensitive.
show.messages	logical, default TRUE for printing messages to the R promt.
.kit.param	data frame, run function on a data frame instead of the parameters.txt file.
debug	logical indicating printing debug information.

Details

The function returns various information for kit and parameters specified in parameters.txt.

Value

vector of data frame with kit information.

Examples

```
# Returns vector of available kits.
getParameter()

# Returns vector of all methods.
getParameter(what="methods")

# Returns methods for specified kit.
getParameter(kit="SGMPlus", what="methods")

# Returns vector of available options.
getParameter(what="options")

# Returns vector of markers for specified kit.
getParameter(kit="SGMPlus", what="Marker")

# Returns data frame of all information for specified kit and method.
getParameter(kit="SGMPlus", method = "Default")
```

---

pcrsim *GUI for PCRsim*

Description

A graphical user interface for simulation of the entire DNA process.

Usage

```
pcrsim(debug = FALSE)
```

Arguments

debug	logical, indicating if debug information should be printed.

Details

This graphical user interface give access to parameters and functions for simulation of the forensic DNA process. Detailes are entered for each sub process organised into tabs. Simulation is performed and the result can be viewed within the GUI, plotted as an electropherogram (EPG), or saved to a text file. The EPG can be saved as an image.

Examples

```
## Not run:
# Open the graphical user interface.
pcrsim()

## End(Not run)
```

---

rmultinomxl *Vectorized Multinomial Distribution For Large Numbers*

Description

Simulates one cycle of the PCR process.

Usage

```
rmultinomxl(n, size, prob, check = TRUE, debug = FALSE)
```

Arguments

n	numeric vector with number of observations.
size	numeric vector specifying the number of molecules going into the PCR cycle.
prob	numeric non-negative matrix of size (x x K) specifying the probability for the K classes.
check	logical for extended checking of parameters.
debug	logical to print debug information.

Details

Generate multinomially distributed random number vectors using `rmultinomial`. Handles integer overflow by splitting numbers in chunks and call `rmultinomial` repeatedly.

Value

matrix with simulated results

`simCE` *CE Simulator*

Description

Simulates the capillary elechtrophoresis (CE) process.

Usage

```
simCE(data, vol = 1, sd.vol = 0, intercept = -7.7662, slope = 0.859,
  sigma = 0.5798, t.intercept = 10.82719, t.slope = 0.9047,
  t.sigma = 0.5951, debug = FALSE)
```

Arguments

`data`	data.frame with simulated data. Preferably output from `simPCR`. Required columns are 'PCR.Vol' and 'PCR.Amplicon'.
`vol`	numeric for the aliquot PCR product for CE analysis.
`sd.vol`	numeric for the standard deviation of `vol`.
`intercept`	numeric for the intercept of the linear model to scale molecules -> rfu.
`slope`	numeric for the slope of the linear model to scale molecules -> rfu.
`sigma`	numeric for the residual standard error of the linear model to scale molecules -> rfu. NB! NOT USED!
`t.intercept`	numeric for the intercept of the linear model to calculate detection threshold (molecules).
`t.slope`	numeric for the slope of the linear model to calculate detection threshold (molecules). NB! NOT USED!
`t.sigma`	numeric for the residual standard error of the linear model to calculate detection threshold (molecules).
`debug`	logical for printing debug information.

Details

Simulates the forensic capillary electrophoresis analysis of PCR product.

Value

data.frame with simulation results in columns 'CE.xxx'.

See Also

`simPCR`

Examples

```
# Create a data frame with a DNA profile.
markers = rep(c("D3S1358","TH01","FGA"), each=2)
alleles = c(15,18,6,10,25,25)
df <- data.frame(Marker=markers, Allele=alleles)

# Simulate profile.
res <- simProfile(data=df, sim=5, name="Test")

# Simulate sample
res <- simSample(data=res, cells=58, sd.cells=0)

# Simulate extraction.
res <- simExtraction(data=res, vol.ex=1, sd.vol=0, prob.ex=1, sd.prob=0)

# Simulate PCR.
res <- simPCR(data=res, kit=NULL, pcr.cyc=30, vol.aliq=1, sd.vol.aliq=0, vol.pcr=25, sd.vol.pcr=0)

# Simulate CE.
res <- simCE(data=res, vol=1, sd.vol=0, intercept=-10.48, slope=0.86, sigma=0.58)
print(res)
```

`simDegradation` *DNA Degradation Simulator*

Description

Simulates the degradation of DNA.

Usage

```
simDegradation(data, kit, deg, quant.target, debug = FALSE)
```

Arguments

`data`	data.frame with simulated data. Preferably output from `simSample`. Required columns are 'Allele', 'DNA', and 'Sim'.
`kit`	character string for STR DNA amplification kit.
`deg`	numeric for the estimated degradation probability (chance per base pair)
`quant.target`	integer defining the quantification target size in base pair.
`debug`	logical TRUE to indicate debug mode.

Details

Simulates the DNA degradation process by calculating the probability that a DNA fragment of the specific size (bp) is complete i.e. not degraded. Then a binomial selection of non-degraded molecules takes place with the previously calculated probability. The number of molecules is taken from the required column 'DNA' which is `floored` to avoid NAs in the `rbinom` function.

Value

data.frame with simulated results in columns 'Deg.Par', 'Deg.Prob', and 'Deg.DNA'.

See Also

simSample

Examples

```
# Simulate profile.
# Get allele frequency database.
require(strvalidator)
db <- strvalidator::getDb(getDb()[2])
# Simulate profile.
res <- simProfile(kit= "ESX17", db=db, sim=10, name="Test")
# Simulate sample.
res <- simSample(data=res, cells=5000)
# Simulate degradation.
res <- simDegradation(data=res, kit="ESX17", deg=0.03, quant.target=80)
print(res)
```

---

simDilution *Serial Dilution Simulator*

Description

Simulates the serial dilution process of a DNA extract.

Usage

```
simDilution(data = NULL, stock.conc = 10, stock.vol = 1000,
  stock.aliq = 5, amount = c(1, 0.1), steps = 3, dilution.factor = 0.5,
  dilution.aliq = 100, pcr.aliq = 17.5, cell.dna = 0.006,
  truncate.top = TRUE, debug = FALSE)
```

Arguments

data	data.frame with simulated data.
stock.conc	numeric for the stock concentration (ng/ul).
stock.vol	numeric for the stock volume (ul).
stock.aliq	numeric for the aliquot volume pipetted from the stock solution (ul).
amount	numeric for the highest target amount in the PCR reaction (ng) or a vector of length two giving the target range c(high, low).
steps	numeric for the number of dilution steps.
dilution.factor	numeric between 0 and 1 for the dilution factor (i.e. 0.5 for a two-fold dilution).

dilution.aliq	numeric for the aliquot pipetted in each serial dilution step (excluding from stock solution).
pcr.aliq	numeric for the aliquot forwarded to PCR (ul).
cell.dna	numeric to indicate the DNA content of a diploid cell in nanograms (ng).
truncate.top	logic if TRUE the high concentrations will be discarded if there are too few simulated samples in 'data'. If FALSE the low concentrations will be discarded.
debug	logical flagging for debug mode.

Details

Simulates the dilution process by binomial selection of molecules from a stock of extracted DNA. Result include columns with number of molecules prior to the binomial selection (Dil.DNA.Pre) the probability used (Dil.Prob), the total volume of each serial dilution prior to removal of aliquot for the subsequent dilution (Dil.Vol.Ser), the number of molecules in each serial dilution prior to removal of aliquot for the subsequent dilution (Dil.DNA.Ser), final volume after serial dilution is complete (Dil.Vol), final number of molecules after serial dilution is complete (Dil.DNA), and target concentration used in calculations (Dil.Conc).

Value

data.frame with simulation results in columns 'Dil.DNA.Pre', 'Dil.Prob', 'Dil.Vol.Ser', 'Dil.DNA.Ser', 'Dil.Vol', 'Dil.DNA', and 'Dil.Conc'. Columns 'Volume' and 'DNA' are added or updated to be used subsequently.

Examples

```
# Create a data frame with a DNA profile.
markers = rep(c("D3S1358","TH01","FGA"), each=2)
alleles = c(15,18,6,10,25,25)
res <- data.frame(Marker=markers, Allele=alleles)

# Simulate profile.
res <- simProfile(data=res, sim=3, name="Test")

# Simulate diploid sample.
res <- simSample(data=res, cells=10000, sd.cells=200)

# Simulate extraction.
res <- simExtraction(data=res, vol.ex=200, sd.vol=10, prob.ex=0.3, sd.prob=0.1)

# Simulate dilution.
res <- simDilution(data=res, amount=1, dilution.factor=0.5)
```

simExtraction *DNA Extraction Simulator*

Description

Simulates the DNA extraction process.

Usage

```
simExtraction(data = NULL, vol.ex = 100, sd.vol = 0, prob.ex = 0.3,
  sd.prob = 0, cell.dna = 0.006, debug = FALSE)
```

Arguments

data	data.frame with simulated data. Preferably output from simSample. Required columns are 'Marker', 'Allele', 'Sim', and 'DNA'.
vol.ex	numeric for the final extraction volume (volume after extraction).
sd.vol	numeric for the standard deviation of vol.ex.
prob.ex	numeric for probability that an allele survives the extraction (extraction efficiency).
sd.prob	numeric for the standard deviation of prob.ex.
cell.dna	numeric to indicate the DNA content of a diploid cell in nanograms (ng).
debug	logical TRUE to indicate debug mode.

Details

Simulates the DNA extraction process by a series of normal distributions. The number of molecules is taken from the required column 'DNA' which is floored to avoid NAs in the rbinom function.

Value

data.frame with simulation results in columns 'Ex.Vol', 'Ex.Prob', Ex.DNA', 'Ex.Conc', and updated 'DNA' and 'Volume' columns (added if needed).

See Also

simSample

Examples

```
# Create a data frame with a DNA profile.
markers = rep(c("D3S1358","TH01","FGA"), each=2)
alleles = c(15,18,6,10,25,25)
df <- data.frame(Marker=markers, Allele=alleles)

# Simulate profile.
res <- simProfile(data=df, sim=3, name="Test")
```

```
# Simulate diploid sample.
res <- simSample(data=res, cells=100, sd.cells=20)

# [OPTIONAL] Simulate degradation.
res <- simDegradation(data=res, kit="ESX17", deg=0.003, quant.target=80)

# Simulate extraction.
res <- simExtraction(data=res, vol.ex=200, sd.vol=10, prob.ex=0.3, sd.prob=0.1)
```

`simNormalize` *Normalization Simulator*

Description

Simulates the normalization process of a DNA extract.

Usage

```
simNormalize(data = NULL, volume = NULL, accuracy = 1,
  target = 0.5/17.5, tolerance = 0.1, multiple = FALSE, debug = FALSE)
```

Arguments

`data`	data.frame with simulated data. Preferably output from `simExtraction`. Required columns are 'Marker', 'Allele', 'Sim', 'Volume', 'Ex.Conc', and 'DNA'.
`volume`	numeric for the final volume after dilution.If NULL it will be taken from column 'Volume'.
`accuracy`	numeric for the pipetting accuracy e.g. minimum pipetting volume.
`target`	numeric for the target concentration.
`tolerance`	numeric for the tolerance around the target concentration e.g. 0.1 is +-10%.
`multiple`	logic if TRUE the function will call itself until `target` is reached. Only the last round of results will be stored in the simulated dataset.
`debug`	logical flagging for debug mode.

Details

Simulates the normalization process by binomial selection of molecules. The average concentration per sample is used to calculate the dilution factor.

Value

data.frame with simulation results in columns 'Norm.Avg.Conc', 'Norm.Vol', 'Norm.Aliq', 'Norm.Aliq.Prob', 'Norm.DNA', 'Norm.Conc', and 'DNA'.

See Also

simExtraction

Examples

```
# Create a data frame with a DNA profile.
markers = rep(c("D3S1358","TH01","FGA"), each=2)
alleles = c(15,18,6,10,25,25)
df <- data.frame(Marker=markers, Allele=alleles)

# Simulate profile.
res <- simProfile(data=df, sim=3, name="Test")

# Simulate diploid sample.
res <- simSample(data=res, cells=10000, sd.cells=200)

# [OPTIONAL] Simulate degradation.
res <- simDegradation(data=res, kit="ESX17", deg=0.003, quant.target=80)

# Simulate extraction.
res <- simExtraction(data=res, vol.ex=200, sd.vol=10, prob.ex=0.3, sd.prob=0.1)

# Simulate normalization.
res <- simNormalize(data=res, volume=100)
```

---

simPCR *PCR Simulator*

Description

Simulates the Polymerase Chain Reaction (PCR) process.

Usage

```
simPCR(data, kit = NULL, method = "DEFAULT", pcr.cyc = 30,
  pcr.prob = 0.8, sd.pcr.prob = 0, stutter = TRUE, stutter.prob = 0.002,
  sd.stutter.prob = 0, stutter.reg = FALSE, vol.aliq = 10,
  sd.vol.aliq = 0, vol.pcr = 25, sd.vol.pcr = 0, debug = FALSE)
```

Arguments

data	data.frame with simulated data. Preferably output from simExtraction. Required columns are 'Marker', 'Allele', 'Sim', Volume', and 'DNA'.
kit	string representing the typing kit used, or data.frame with kit characteristics. Provides locus specific PCR efficiency and stutter probabilities. If NULL pcr.prob and stutter.prob will be used for all loci.
method	string representing the method of the specified kit.
pcr.cyc	numeric for the number of PCR cycles.

pcr.prob	numeric for the PCR efficiency (probability amplifying a DNA molecule). Only used if kit is NULL.
sd.pcr.prob	numeric for the standard deviation of pcr.prob.
stutter	logical to simulate stutters.
stutter.prob	numeric for the probability generating a stutter. Only used if kit is NULL.
sd.stutter.prob	numeric for the standard deviation of stutter.prob.
stutter.reg	logical to use regression for stutter probability.
vol.aliq	numeric for the aliquot extract forwarded to PCR.
sd.vol.aliq	numeric for the standard deviation of vol.aliq.
vol.pcr	numeric for the total PCR reaction volume.
sd.vol.pcr	numeric for the standard deviation of vol.pcr.
debug	logical to print debug information.

Details

Simulates the PCR process by a series of binomial distributions, or multinomial distributions if stutter=TRUE. The PCR probability/ efficiency can be specified globally or per locus. Probability of stutter formation can be specified globally, per locus, or per locus and allele size.

Value

data.frame with simulation results in columns 'PCR.Pip.Vol', 'PCR.Aliq.Prob', 'PCR.DNA', 'PCR.Vol', 'PCR.Prob', 'PCR.Prob.Stutter', 'PCR.Amplicon', 'PCR.Stutter.1', 'PCR.Stutter.2', and updated 'DNA' column (added if needed).

See Also

simExtraction

Examples

```
# Create a data frame with a DNA profile.
markers = rep(c("D3S1358","TH01","FGA"), each=2)
alleles = c(15,18,6,10,25,25)
df <- data.frame(Marker=markers, Allele=alleles)

# Simulate profile.
smpl <- simProfile(data=df, sim=10)

# Simulate sample.
smpl <- simSample(data=smpl, cells=1000, sd.cells=200)

# Simulate extraction.
smpl <- simExtraction(data=smpl, vol.ex=200, sd.vol=10, prob.ex=0.6, sd.prob=0.1)

# Simulate PCR with 95% PCR efficiency and 0.2% stutter probability for all loci.
res <- simPCR(data=smpl, pcr.prob=0.95, pcr.cyc=30, vol.aliq=10,
```

```
 sd.vol.aliq=0.1, vol.pcr=25, sd.vol.pcr=1)

# Simulate PCR with locus specific PCR efficiency and stutter probability.
res <- simPCR(data=smpl, kit="ESX17", pcr.cyc=30, vol.aliq=10,
 sd.vol.aliq=0.1, vol.pcr=25, sd.vol.pcr=1)

# Simulate PCR with locus specific PCR efficiency and stutter probability.
res <- simPCR(data=smpl, kit="ESX17", pcr.cyc=30, vol.aliq=10,
 sd.vol.aliq=0.1, vol.pcr=25, sd.vol.pcr=1, stutter.reg=TRUE)
```

`simProfile` *DNA Profile Simulator*

Description

Simulates DNA profiles.

Usage

```
simProfile(data = NULL, kit = NULL, sim = 1, name = NULL, db = NULL,
  debug = FALSE)
```

Arguments

`data`	data.frame with columns 'Marker' and 'Allele' (creates fixed profiles).
`kit`	character string to specify the typing kit (if kit=NULL all markers in db will be used).
`sim`	integer to specify the number of replicates or random profiles.
`name`	character string giving the sample base name ('Sim' is appended).
`db`	data.frame with the allele frequency database (used if data=NULL to create random profiles).
`debug`	logical TRUE to indicate debug mode.

Details

There are three ways to create DNA profiles: 1) Simulate DNA profiles of the selected kit using allele frequencies from the provided db. 2) Simulate DNA profiles using all available markers and allele frequencies from the provided db. 3) Provide a data.frame with a fixed DNA profile. `sim` random profiles or `sim` replicates are created. The resulting data.frame can be used as input to `simSample`. NB! Homozygous alleles must be specified two times e.g. 16, 16.

Value

data.frame with columns 'Sample.Name', 'Marker', 'Allele', 'Sim'.

See Also

`simSample`

Examples

```
# Create a data frame with a DNA profile.
markers = rep(c("D3S1358","TH01","FGA"), each=2)
alleles = c(15,18,6,10,25,25)
df <- data.frame(Marker=markers, Allele=alleles)

# Simulate sample
res <- simProfile(data=df, sim=10, name="Test")
print(res)
```

`simSample` *Forensic Sample Simulator*

Description

Simulates the DNA content in forensic stain.

Usage

```
simSample(data, cells = NULL, sd.cells = 0, conc = NULL, sd.conc = 0,
  vol = NULL, sd.vol = 0, cell.dna = 0.006, haploid = FALSE,
  kit = NULL, slope = NULL, intercept = NULL, debug = FALSE)
```

Arguments

`data`	data.frame with columns 'Marker', 'Allele', and 'Sim' defining the DNA profiles and simulation id (counter). Preferably output from `simProfile`.
`cells`	integer for the estimated number of cells.
`sd.cells`	numeric for the standard deviation of `cells`.
`conc`	numeric for the estimated DNA concentration.
`sd.conc`	numeric for the standard deviation of `conc`.
`vol`	numeric for the estimated sample volume.
`sd.vol`	numeric for the standard deviation of `vol`.
`cell.dna`	numeric to indicate the DNA content of a diploid cell in nanograms (ng).
`haploid`	logical TRUE to indicate haploid cells.
`kit`	character string defining the DNA typing kit used to calculate allele size (used to calculate allele sizes needed for the regression option i.e. `slope` and `intercept`).
`slope`	numeric from regression of log concentration by fragment size (bp).
`intercept`	numeric from regression of log concentration by fragment size (bp).
`debug`	logical TRUE to indicate debug mode.

Details

Simulates the number of DNA molecules in a forensic stain either from: 1) An estimate of the number of cells in the stain. 2) The DNA concentration. 3) The slope and intercept values as obtained from log-linear regression of DNA concentration by size in basepair. The regression emulates degradation and should not be used together with simulation of degradation using simDegradation. Some parameters accept vectors so that simulated samples can have different number of cells and be a mixture of haploid and diploid samples (see examples). Note 1: Number of cells can be decimal values since it is an estimate. Note 2: Number of cells will always be integer if haploid=TRUE because binomial selection require integer values. Note 3: To get the same total amount of DNA in samples with diploid and haploid cells. the parameter for haploid cells must be: cells = 2 * number_of_diploid_cells NB! Important that each marker has two rows (i.e. homozygotes is e.g. 16, 16).

Value

data.frame with simulated result in columns 'Cells'.

See Also

simProfile

Examples

```
# Create a data frame with a DNA profile.
markers = rep(c("D3S1358","TH01","FGA"), each=2)
alleles = c(15,18,6,10,25,25)
df <- data.frame(Marker=markers, Allele=alleles)
# Simulate profile.
prof <- simProfile(data=df, sim=3, name="Test")

# Simulate diploid sample.
res <- simSample(data=prof, cells=100, sd.cells=20)
print(res)

# Simulate haploid sample.
res <- simSample(data=prof, cells=100, sd.cells=20, haploid=TRUE)
print(res)

# Simulate haploid sample from concentration.
res <- simSample(data=prof, conc=0.02, sd.conc=0.001, vol=100, haploid=TRUE)
print(res)

# Simulate sample from slope and intercept.
res <- simSample(data=prof, vol=100, slope=-0.01, intercept=0.20, kit="SGMPlus")
print(res)

# Simulate mixture of diploid and haploid sample types of two concentrations.
res <- simSample(data=prof, cells=c(1000,1000,250), haploid=c(FALSE,TRUE,FALSE))
print(res)
```

Index