Package 'rsalvador'

April 20, 2014

Version 1.0

Date 2014-04-20

Title rSalvador: An R Tool for the Luria-Delbruck Fluctuation Assay

Author Qi Zheng <qzheng@sph.tamhsc.edu>

Maintainer Qi Zheng <qzheng@sph.tamhsc.edu>

Depends R (>= 2.15.0), hypergeo, gdata

Description Routines for inferring microbial mutation rates from fluctuation assay data. rSalvador can compute maximum likelihood estimates and likelihood ratio-based confidence intervals, adjusting for plating efficiency when necessary. In addition to the usual continuous-time Lea-Coulson mutation model, rSalvador can accommodate Haldane's discrete-time mutation model. Furthermore, rSalvador can compute mutant cell distribution functions, plot log-likelihood functions and simulate fluctuation experiments.

LazyData yes

License GPL (>= 2)

R topics documented:

`rsalvador-package` *rSalvador: An R Tool for the Luria-Delbruck Fluctuation Assay.*

Description

Adapted from SALVADOR 2.3 (Zheng, 2008)

Details

Package:	rSalvador
Type:	Package
Version:	1.0
Date:	2014-04-20
License:	GPL 2.0

Note

Eric H. Zheng provided technical support.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, SALVADOR 2.3: A Tool for Studying Mutation Rates, published in September 2008, available at http://library.wolfram.com/infocenter/MathSource/7082.

`cairns.foster.data` *Experimental Data from Cairns and Foster*

Description

Well-known data, analyzed by Rosche and Foster (2000).

Details

Experiment described in detail in Cairns and Foster (1991), data first appeared in Rosche and Foster (2000) with the maximum likelihood estimate of m explicitly given.

Source

WA Rosche and PL Foster, Determining mutation rates in bacterial population, Methods 20: 4-17 (2000)

References

J Cairns and PL Foster, Adaptive reversion of a frameshift mutation in Escherichia coli, Genetics 128:695-701 (1991)

Examples

```
newton.LD(cairns.foster.data,show.iter=TRUE)
```

---

confint.Haldane *Confidence Intervals for Mutation Rates Under the Haldane Model*

Description

It uses an algorithm described in Zheng (2007) to compute a confidence interval for mu, the probability of mutation per cell division under the Haldane model.

Usage

```
confint.Haldane(data, g, N0 = 1, alpha = 0.05, tol = 1e-06, init.mu = 0,
init.lower = 0, init.upper = 0, max.iter = 30, show.iter = FALSE)
```

Arguments

data	a vector of experimental data (non-negative integers).
g	number of generations.
N0	initial number of nonmutant cells.
alpha	level of the confidence interval.
tol	tolerance parameter to control the iteration process.
init.mu	an initial guess for mu.
init.lower	an initial guess for lower limit of m, rarely needed.
init.upper	an initial guess for upper limit of m, rarely needed.
max.iter	maximum number of iterations.
show.iter	to show the iteration progress.

Details

The algorithm is described in Zheng (2007).

Value

a confidence interval for the mutation rate mu.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, (2007). On Haldane's formulation of Luria and Delbruck's mutation model, Mathematical Biosciences 209:500-513.

See Also

confint.LD confint.LD.plating

Examples

```
confint.Haldane(demerec.data,g=25,N0=90,show.iter=TRUE)
```

`confint.LD` *Likelihood Ratio-Based Confidence Interval under the Luria-Delbruck Model*

Description

This function employs an algorithm described in Zheng (2005) to compute a likelihood ratio-based confidence interval for m, the expected number of mutations per test tube.

Usage

```
confint.LD(data, alpha = 0.05, phi = 1, tol = 1e-06, init.m = 0,
 init.lower = 0, init.upper = 0, max.iter = 30, show.iter = FALSE)
```

Arguments

`data`	a vector of numbers of mutant colonies
`alpha`	significance level
`phi`	this parameter is defined by phi=1-N0/Nt, is usually assumed to be 1.0.
`tol`	tolerance parameter
`init.m`	an initial guess for m, rarely needed
`init.lower`	an initial guess for lower limit of m, rarely needed
`init.upper`	an initial guess for upper limit of m, rarely needed
`max.iter`	maximum number of iterations
`show.iter`	to show iteration progress

Details

The algorithm is described in Zheng (2005).

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, (2005). New algorithms for Luria-Delbruck fluctuation analysis, Mathematical Biosciences 196:198-214 (2005).

See Also

confint.LD.plating

Examples

```
confint.LD(demerec.data,show.iter=TRUE)
```

---

confint.LD.plating *Confidence Intervals for m that Accounts for Plating Efficiency*

Description

This function computes likelihood ratio-based confident intervals for m that accounts for plating efficiency. The algorithm was described in Zheng (2008).

Usage

```
confint.LD.plating(data, e, alpha = 0.05, tol = 1e-06, init.m = 0,
 init.lower = 0, init.upper = 0, max.iter = 30, show.iter = FALSE)
```

Arguments

data	A vector of experimental data, all integers.
e	Plating efficiency e (0<e<1)
alpha	A 1-alpha confidence interval is to be constructed.
tol	Tolerance parameter to control the number of iterations.
init.m	An initial guess for m can be supplied, but is rarely needed.
init.lower	An initial guess for the lower limit of the confidence interval, but is rarely needed.
init.upper	An initial guess for the upper limit of the confidence interval, but is rarely needed.
max.iter	Maximum number of iterations allowed.
show.iter	To view the iteration process.

Value

A pair of positive real numbers.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng. A note on plating efficiency in fluctuation experiments, Mathematical Biosciences, 216:150-153 (2008).

See Also

confint.LD

`demerec.data` *Classic Experimental Data of Demerec*

Description

This data set came from one of the earliest fluctuation experiments performed by one of the pioneers in the field. It has been cited in the literature numerous times.

Source

M. Demerec, Production of staphylococcus strains resistant to various concentrations of penicillin, Proc. Natl. Acad. Sci. USA 31 (1945) 16-24.

Examples

```
newton.LD(demerec.data,show.iter=TRUE)
```

`export.text.data` *Exporting Data to an External Text File*

Description

It is sometimes desirable to save experimental data already in rSalvador to an external text file. For example, data imported by import.excel.data may later be exported as a text file so that other programs can use the data.

Usage

```
export.text.data(filename, data)
```

Arguments

`filename`	Name of the text file to be written.
`data`	A vector of experimental data to be exported.

Author(s)

Qi Zheng <qzheng@sprh.tamhsc.edu>

See Also

import.text.data

Examples

```
export.text.data('testing.txt',demerec.data)
```

`import.excel.data` *Importing Data from an Excel File*

Description

Biologists often keep their experimental results in Excel files. In addition to the numbers of mutant cells, such files usually contain other experimental details. rSalvador requires excel files to satisfy two conditions. First, the first row of the file should be reserved as a header, and second, the first non-blank column should be reserved to keep the numbers of mutant cells that rSalvador intends to import. The parameter col (which has a default value of 1) can be used to choose which column of the file is to be imported.

Usage

```
import.excel.data(filename, col = 1)
```

Arguments

`filename`	the data file to be imported.
`col`	the desired column to be imported.

Value

A vector of non-negative integers.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

See Also

import.text.data

`import.text.data` *Importing Data from a Text File*

Description

The text data file to be imported can have several header lines at the beginning, followed by a single-column of experimental data. The parameter jump specifies the number of header lines.

Usage

```
import.text.data(filename, jump = 1, col = 1)
```

Arguments

filename	Experimental data should be listed in a single column, but the first few lines can be the experimenter's note.
jump	Number of lines to skip.
col	Which column contains data.

Value

A vector of non-negative integers imported form the text file.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

---

log.likelihood.Haldane

Computing the Log-Likelihood Function Under the Haldane Model

---

Description

This function computes the log-likelihood function for data under the Haldane model. The algorithm is based on Zheng (2007).

Usage

```
log.likelihood.Haldane(data, g, mu, N0 = 1)
```

Arguments

data	A vector of data.
g	Number of generations.
mu	The mutation rate.
N0	Initial number of nonmutant cells.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng (2007) On Haldane's formulation of Luria and Delbruck's mutation model, Mathematical Biosciences 209:500-513.

See Also

log.likelihood.LD, log.likelihood.LD.plating

Examples

```
log.likelihood.Haldane(niccum.data,25,1e-6)
```

log.likelihood.LD *Computing the Log-Likelihood Function Under the Luria-Delbruck Model*

Description

This function computes the log-likelihood function for data under the Lea-Coulson formulation of the Luria-Delbruck mutation model.

Usage

```
log.likelihood.LD(data, m, phi = 1)
```

Arguments

data	A vector of data.
m	The parameter m is regarded as the argument of the log-likelihood function.
phi	This parameter is defined as phi=1-N0/Nt.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

See Also

log.likelihood.LD.plating

Examples

```
log.likelihood.LD(demerec.data,m=8.5)
```

log.likelihood.LD.plating *Computing the Log-Likelihood Function That Adjusts for Plating Efficiency*

Description

This function computes the log-likelihood function that adjusts for plating efficiency. The parameter m is regard as the argument of the log-likelihood function.

Usage

```
log.likelihood.LD.plating(data, m, e)
```

Arguments

data	A vector of experimental data.
m	A real positive number as a given value of the parameter m.
e	plating efficiency.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

See Also

log.likelihood.LD

---

luria.16.data *Classic Experimental Data from Luria and Delbruck*

Description

Data taken from Luria and Delbruck (1943) and analyzed in Zheng (2008).

Details

In this experiment, each test tube had a 0.2-ml culture, but only a portion of 0.08 ml was plated. Therefore, the plating efficiency was e=0.4.

References

S.E. Luria, M. Delbruck, Mutations of bacteria from virus sensitivity to virus resistance, Genetics 28 (1943) 491-511.

Q. Zheng. A note on plating efficiency in fluctuation experiments, Mathematical Biosciences, 216 (2008) 150-153.

Examples

```
newton.LD.plating(luria.16.data, e=0.4, show.iter = TRUE)
```

---

newton.Haldane *Computing MLE of the Mutation Rate Under the Haldane Model*

Description

This function computes maximum likelihood estimates of mu, the probability of a mutation per cell division, under the Haldane model. The Haldane model was described by Sarkar (1991), the maximum likelihood estimate of mu is computed using a Newton-Raphson type iterative algorithm described in Zheng (2007).

Usage

```
newton.Haldane(data, g, N0 = 1, tol = 1e-08, init.mu = 0, max.iter = 30,
 show.iter = FALSE)
```

Arguments

data	a data vector, containing numbers of mutant cells in all tubes.
g	the number of generations.
N0	the number of initial cells.
tol	tolerance parameter to control convergence.
init.mu	an initial guess of mu can be supplied.
max.iter	maximum number of iterations allowed.
show.iter	exhibition of the intermediate values of m during the iteration process.

Value

A positive number.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

S. Sarkar (1991) Haldane's solution of the Luria-Delbruck distribution, Genetics 127, 257-261.

Q. Zheng (2007) On Haldane's formulation of Luria and Delbruck's mutation model, Mathematical Biosciences 209:500-513.

See Also

newton.LD, newton.LD.plating

Examples

```
newton.Haldane(niccum.data,g=25,show.iter=TRUE)
```

---

newton.LD *Computing MLE of m Under the Luria-Delbruck Model*

Description

This function computes maximum likelihood estimates of m, the expected number of mutations per culture, using a Newton-Raphson type algorithm described in Zheng (2005).

Usage

```
newton.LD(data, phi = 1, tol = 1e-08, init.m = 0, max.iter = 30,
 show.iter = FALSE)
```

Arguments

data	a data vector, containing numbers of mutant cells in all test tubes.
phi	this parameter is defined by 1-N0/Nt, usually phi is taken as unity.
tol	tolerance parameter to control convergence.
init.m	an initial guess of m, it is rarely needed.
max.iter	maximum number of iterations.
show.iter	exhibition of the intermediate values of m during the iteration process.

Value

A positive number.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, (2005). New algorithms for Luria-Delbruck fluctuation analysis, Mathematical Biosciences 196:198-214 (2005).

See Also

newton.LD.plating

Examples

```
newton.LD(demerec.data,show.iter=TRUE)
```

---

newton.LD.plating *Computing Maximum Likelihood Estimate of m that Adjusts for Plating Efficiency*

Description

This function computes the m.l.e. of m (the expected number of mutations) when plating efficiency is imperfect. It employs a Newton-Raphson type algorithm described in Zheng (2008).

Usage

```
newton.LD.plating(data, e, tol = 1e-08, init.m = 0, max.iter = 30,
 show.iter = FALSE)
```

Arguments

data	a vector of numbers of mutant colonies in a fluctuation experiment.
e	plating efficiency assumed to be known.
tol	tolerance parameter to control the number of iterations.
init.m	an initial guess about m can be specified, but is rarely needed.
max.iter	maximum number of iterations allowed.
show.iter	to view the iteration process.

Value

A positive number.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, (2008) A note on plating efficiency in fluctuation experiments, Mathematical Biosciences, 216:150-153.

See Also

newton.LD

`niccum.data` *An Experiment Reported by Niccum et al.*

Description

Data taken from Niccum et al. (2012).

Details

This data set is the Time 1 part of Experiment A of Niccum et al. (2012). The estimated number of viable cells is 31350000, close to 33554432. As N0=1 was assumed, the number of generations is therefore g=25.

References

Niccum, BA, Poteau, R, Hamman, GE, Varada, JC, Dshalalow, JH, Sinden, RR, 2012. On an unbiased and consistent estimator for mutation rates, Journal of Theoretical Biology 300, 360-367.

Examples

```
newton.Haldane(niccum.data,g=25)
```

`plot.likelihood.Haldane` *Plotting the Log-Likelihood Function Under the Haldane Model*

Description

This function plots the log-likelihood function of data under the Haldane model. The argument of the log-likelihood function is the mutation rate mu. It is based on the algorithm described in Zheng (2007).

Usage

```
plot.likelihood.Haldane(data,g, init.mu=0,  mu.low = -1, mu.up = -1,
 plot.pts = 30, lik.col = "black", mle.col = "red", title = "",
 x.lab = "Value of mu", y.lab = "Log-likelihood", show.secant = TRUE)
```

Arguments

`data`	A vector of experimental data (non-negative integers).
`g`	number of generations.
`init.mu`	a user specified initial guess of the mutation rate mu.
`mu.low`	plot.likelihood.Haldane plots the log-likelihood function for mu between mu.low and mu.up.
`mu.up`	see above.
`plot.pts`	number of points used to plot the log-likelihood functions. A larger number of points yields a smoother graph, but takes more computing time.
`lik.col`	color for the log-likelihood function.
`mle.col`	color for the vertical bar that marks the maximum likelihood estimate of m.
`title`	title to be given to the graph.
`x.lab`	x label.
`y.lab`	y label.
`show.secant`	A secant line will be drawn at the maximum of the log-likelihood function.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, On Haldane's formulation of Luria and Delbruck mutation model, Mathematical Biosciences 209 (2007) 500-513.

See Also

plot.likelihood.LD, plot.likelihood.LD.plating

Examples

```
plot.likelihood.Haldane(niccum.data,g=25)
```

`plot.likelihood.LD` *Plotting the Log-Likelihood Function under the Luria-Delbruck Model*

Description

This function plots the log-likelihood function of data under the Luria-Delbruck model, using the stochastic formulation first proposed by Lea and Coulson (1948). A history of the Lea-Coulson formulation is given in Zheng (1999).

Usage

```
plot.likelihood.LD(data, m.low = -1, m.up = -1, plot.pts = 30,
  lik.col = "black", mle.col = "red", title = "", x.lab = "Value of m",
  y.lab = "Log-likelihood", show.secant = TRUE)
```

Arguments

`data`	A vector of experimental data (non-negative integers).
`m.low`	plot.likelihood.LD plots the log-likelihood function for m ranging from between m.low to m.up.
`m.up`	see above.
`plot.pts`	number of points used to plot the log-likelihood functions. A larger number of points yields a smoother graph, but takes more computing time.
`lik.col`	color for the log-likelihood function.
`mle.col`	color for the vertical bar that marks the maximum likelihood estimate of m.
`title`	title to be given to the graph.
`x.lab`	x label.
`y.lab`	y label.
`show.secant`	A secant line will be drawn at the maximum of the log-likelihood function.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, Progress of a half century in the study of the Luria-Delbruck distribution, Mathematical Biosciences 162 (1999) 1-32.

E.A. Lea and C.A. Coulson, The distribution of the numbers of mutants in bacterial populations. J. Genetics 49 (1949) 264-285.

See Also

plot.likelihood.LD.plating

Examples

```
plot.likelihood.LD(demerec.data)
```

`plot.likelihood.LD.plating` *Plotting the Log-Likelihood Function that Adjusts for Plating Efficiency*

Description

This functions plots the log-likelihood function that adjusts for plating efficiency. The plating efficiency e is assumed to be known, the parameter m is regarded as the argument of the log-likelihood function.

Usage

```
plot.likelihood.LD.plating(data, e, m.low = -1, m.up = -1, plot.pts = 30,
  lik.col = "black", mle.col = "red", title = "",
  x.lab = "Value of m", y.lab = "Log-likelihood", show.secant = TRUE)
```

Arguments

`data`	A vector of experimental data (non-negative integers).
`e`	Plating efficiency.
`m.low`	It plots the log-likelihood function for m ranging from m.low to m.up.
`m.up`	see above.
`plot.pts`	number of points used to plot the log-likelihood functions. A larger number of points yields a smoother graph, but takes more computing time.
`lik.col`	color for the log-likelihood function.
`mle.col`	color for the vertical bar that marks the maximum likelihood estimate of m.
`title`	title to be given to the graph.
`x.lab`	x label.
`y.lab`	y label.
`show.secant`	A secant line will be drawn at the maximum of the log-likelihood function.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

See Also

plot.likelihood.LD

Examples

```
plot.likelihood.LD.plating(luria.16.data,e=0.4)
```

`prob.Haldane` *Computing the Haldane Distribution*

Description

The Haldane mutation model is described in Sarkar (1991). Under the Haldane model, cell division is synchronized. When a nonmutant cell divides, the probability is mu that one daughter cell will be a mutant cell. This function computes the Haldane distribution using an algorithm described in Zheng (2007).

Usage

```
prob.Haldane(gen, mu, n, N0)
```

Arguments

`gen`	number of generations.
`mu`	the mutation rate.
`n`	It calculates the first n+1 probabilities, starting from the zeroth probability.
`N0`	The number of initial nonmutant cells.

Value

A vector contain the n+1 probabilities.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

S. Sarkar (1991) Haldane's solution of the Luria-Delbruck distribution, Genetics 127, 257-261.

Q. Zheng, On Haldane's formulation of Luria and Delbruck mutation model, Mathematical Biosciences 209 (2007) 500-513.

See Also

prob.LD

Examples

```
prob.Haldane(12,0.0003,20,1)
```

`prob.LD` *Computing the Luria-Delbruck Mutant Cell Distribution*

Description

This function computes the Luria-Delbruck distribution function. This particular version of the Luria-Delbruck distribution was first proposed by Lea and Coulson (1949), which was not an exact form of the intended distribution. Bartlett later (see, e.g. Bartlett, 1955) modified this distribution by introducing the parameter phi, making the distribution exact. A historical account of this distribution was given by Zheng (1999). The algorithm for computing this distribution was given by Ma et al. (1992), although the parameter phi was not considered at the time.

Usage

```
prob.LD(m, phi, k)
```

Arguments

`m`	expected number of mutations.
`phi`	This parameter is defined as 1-N0/Nt.
`k`	It computes the first k+1 probabilities, starting with the zeroth probability.

Details

Most implementations did not include the clock parameter phi.

Value

A vector of the k+1 probabilities.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

M.S. Bartlett, An Introduction to Stochastic Processes, Cambridge University Press, London, 1955 (pp.132-135).

D.E. Lea and C.A. Coulson, The distribution of the numbers of mutants in bacterial populations, J. Genetics 49 (1949) 264-285.

W.T. Ma, G. vH. Sandri, S. Sarkar, Analysis of the Luria-Delbruck distribution using discrete convolution powers, J. Appl. Prob. 29 (1992) 255-267.

Q. Zheng, Progress of a half century in the study of the Luria-Delbruck distribution, Mathematical Biosciences 162 (1999) 1-32.

See Also

prob.LD.plating

Examples

```
prob.LD(5.8,1,12)
```

`prob.LD.plating` *Calculating the Luria-Delbruck Mutant Cell Distribution that Adjusts for Plating Efficiency*

Description

This function computes the mutant cell distribution that adjusts for plating efficiency. Several authors contributed to the computation of this distribution, among them are Armitage (1952), Stewart (1991) and Jones (1994). Inspired by these important results, Zheng (2008) proposed a computationally feasible algorithm, which prob.LD.plating adopts.

Usage

```
prob.LD.plating(m, e, n)
```

Arguments

`m`	the expected number of mutations per culture.
`e`	plating efficiency ranging between 0 and 1.
`n`	the function computes the first n+1 probabilities, i.e., P0, P1, ..., Pn.

Value

A vector of the first n+1 probabilities.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

P. Armitage, The statistical theory of bacterial populations subject to mutation, J. Royal Statistical Society, ser. B, 14:1-44, 1952.

M. E. Jones, Luria-Delbruck fluctuation experiments; Accounting simultaneously for plating efficiency and differential growth rate, Journal of Theoretical Biology, 166: 355-363, 1994.

F.M. Stewart, Fluctuation analysis: the effect of plating efficiency, Genetica 54: 51-55, 1991.

Q. Zheng, A note on plating efficiency in fluctuation experiments, Mathematical Biosciences, 216:150-153, 2008.

See Also

prob.LD

Examples

```
prob.LD.plating(5.8,0.2,20)
```

simu.cultures *Simulating Fluctuation Experiments*

Description

This function simulates the numbers of mutant cells in all cultures prior to plating. The mutation model used here was proposed by Mandelbrot (1966) and Koch (1974). Under this model, the nonmutant cell growth rate, b1, can differ from that of mutant cells, b2. The expected number of mutations is m=mu*(Nt-N0)/b1. The simulation algorithm is described in Zheng (2002).

Usage

```
simu.cultures(n, mu, b1, b2, N0, Nt)
```

Arguments

n	total number of tubes in an experiment.
mu	the mutation rate, not the expected number of mutations.
b1	wild-type cell growth rate.
b2	mutant cell growth rate.
N0	initial number of wild-type cells in a tube.
Nt	final number of wild-type cells in a tube.

Value

A vector of n non-negative integers.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

A.L. Koch, Mutation and growth rates from Luria-Delbruck fluctuation tests, Mutation Research 95:129-143 (1982).

B. Mandelbrot, A population birth-and-mutation process, I: Explicit distributions for the number of mutants in an old culture of bacteria. J. Appl. Prob. 11:437-444 (1974).

Q. Zheng. Statistical and algorithmic methods for fluctuation analysis with SALVADOR as an implementation. Mathematical Biosciences, 176:237-252 (2002).

Examples

```
simu.cultures(30,10^-8,1,1,20,5*10^8)
```

`simu.Haldane` *Simulating a Fluctuation Experiment Under the Haldane Model*

Description

This function simulates the numbers of mutant cells in a fluctuation experiment under the Haldane model as described in Sarkar (1991) and Zheng (2007).

Usage

```
simu.Haldane(gen, mu, culture)
```

Arguments

`gen`	The number of generations of cell division.
`mu`	The mutation rate
`culture`	The total number of cultures in an experiment.

Value

A vector of non-negative integers.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

S. Sarkar (1991) Haldane's solution of the Luria-Delbruck distribution, Genetics 127, 257-261.

Q. Zheng (2007) On Haldane's formulation of Luria and Delbruck's mutation model, Mathematical Biosciences 209:500-513.

See Also

newton.LD, newton.LD.plating, simu.Kimmel.

Examples

```
simu.Haldane(9,0.03,22)
```

simu.Haldane2 — *Simulating the Numbers of Mutant Cells for the Last Two Generations under the Haldane Model*

Description

This functions is the same as simu.Haldane, except that it returns the numbers of mutant cells for the last two generations.

Usage

```
simu.Haldane2(gen, mu, culture = 1)
```

Arguments

gen	The last generation.
mu	The mutation rate.
culture	The Number of cultures to be simulated.

Value

A pair of non-negative integer vectors.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

See Also

simu.Kimmel2

Examples

```
simu.Haldane2(9,0.08,20)
```

simu.Kimmel — *Simulating a Fluctuation Experiment Under a variation of the Haldane Model*

Description

This function simulates the numbers of mutant cells in a fluctuation experiment under a variation of the Haldane model. Unlike the original Haldane model, this model allows both daughter cells to be mutant cells when a nonmutant cell divides. (Each daughter cell has a probability mu to be a mutant cell.) This model was studied by Kimmel and Axelrod (1994).

Usage

```
simu.Kimmel(gen, mu, culture)
```

Arguments

gen	The number of generations.
mu	The mutation rate.
culture	The total number of cultures in an experiment.

Value

A vector of non-negative integers.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

M. Kimmel, D.E. Axelrod, 1994. Fluctuation test for two-stage mutations: application to gene amplification, Mutation Research 306, 45-60.

See Also

simu.Haldane

Examples

```
simu.Kimmel(9,0.03,22)
```

---

simu.Kimmel2 *Simulating the Numbers of Mutant Cells for the Last Two Generations Under the Kimmel Model*

Description

This function is the same as simu.Kimmel, except that it returns the numbers of mutant cells for the last two generations.

Usage

```
simu.Kimmel2(gen, mu, culture = 1)
```

Arguments

gen	The last generation.
mu	The mutation rate.
culture	The number of cultures to be simulated.

Value

A pair of non-negative integer vectors.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

See Also

simu.Haldane2

Examples

```
simu.Kimmel2(9,0.08,20)
```

Index