

Package ‘rsalvador’

June 16, 2015

Version 1.2

Date 2015-06-16

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 R routines for inferring microbial mutation rates from fluctuation assay data. rSalvador can compute maximum likelihood estimates and likelihood ratio-based confidence intervals, adjusting for relative fitness or plating efficiency when necessary. rSalvador can also be used to compare mutation rates. 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	1
boshoff.data	2
cairns.foster.data	2
compare.LD	3
confint.Haldane	4
confint.LD	5
confint.LD.plating	6
confint.MK	7
crane.data	8
demerec.data	8
export.text.data	8
ford.data	9
import.excel.data	9
import.text.data	10
LD.p0.est	11
likely.average	11

log.likelihood.Haldane	13
log.likelihood.LD	13
log.likelihood.LD.plating	14
LRT.MK	15
luria.16.data	16
newcombe.data	16
newton.Haldane	17
newton.LD	18
newton.LD.plating	19
newton.MK	20
niccum.data	21
plot.likelihood.Haldane	21
plot.likelihood.LD	22
plot.likelihood.LD.plating	23
plot.likelihood.MK	24
prob.Haldane	25
prob.LD	26
prob.LD.plating	27
prob.MK	28
simu.Bartlett	29
simu.cultures	30
simu.Haldane	31
simu.Haldane2	32
simu.Kimmel	32
simu.Kimmel2	33
wh.data	34

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: 1.2
Date: 2015-06-16
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>.

boshoff.data

Experimental Data by Boshoff et al.

Description

Data from 6 fluctuation experiments by Boshoff et al. (2003). The final cell population sizes N_t are: 7.2e8, 3.3e8, 1.02e9, 7.65e8, 2.1e9 and 1.9e9. According to the authors, some large mutants counts are not very reliable. Data are from Supplementary Table of Boshoff (2003).

References

HIM Boshoff, MB Reed, CE Barry III, V Mizrahi, DnaE2 polymerase contributes to in vivo survival and the emergence of drug resistance in Mycobacterium tuberculosis, Cell 113 (2003) 183-193.

Examples

```
newton.LD(unlist(boshoff.data[1]))/(7.2e8)
```

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)
```

compare.LD

Compare mutation rates with a common Nt

Description

This function compares the expected number of mutations per test tube (m) between two fluctuation experiments. Because a common Nt is assumed, it compares the mutation rates between the two experiments.

Usage

```
compare.LD(x1, x2, init.m0 = 0, init.m1 = 0, init.m2 = 0, phi = 1)
```

Arguments

x1	a data vector representing mutant counts in the first experiment.
x2	a data vector representing mutant counts in the second experiment.
init.m0	initial value for estimating a common m shared by the two experiments.
init.m1	initial value for estimating m for the first experiment.
init.m2	initial value for estimating m for the second experiment.
phi	the common clock parameter, often set to unity.

Value

a pair of positive real numbers: (chi-squared test statistic, p-value).

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q Zheng, Methods for comparing mutation rates using fluctuation assay data, Mutat. Res. Fundam. Mol. Mech. Mutagen. 777 (2015) 20-22.

Examples

```
x=simu.cultures(15,1e-7,1,1,5,1e7)
y=simu.cultures(15,3e-7,1,1,5,1e7)
compare.LD(x,y)
```

confint.Haldane	<i>Confidence Intervals for Mutation Rates Under the Haldane Model</i>
-----------------	--

Description

It uses an algorithm described in Zheng (2007) to compute a confidence interval for μ , 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 μ .
init.lower	an initial guess for lower limit of μ , rarely needed.
init.upper	an initial guess for upper limit of μ , 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 μ .

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	<i>Likelihood Ratio-Based Confidence Interval under the Luria-Delbruck Model</i>
------------	--

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 $\phi=1-N_0/N_t$, 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

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

confint.MK	<i>Likelihood-Based Confidence Interval under the Mandelbrot-Koch Model</i>
------------	---

Description

This function computes confidence intervals for m under the Mandelbrot-Koch model that allows for differential growth between mutants and nonmutants. A value of the relative fitness w is obtained from a fitness assay. The algorithm used is a specialization of a more general algorithm described in Zheng (2005).

Usage

```
confint.MK(data, w = 1, alpha = 0.05, tol = 1e-08, init.m = 0,
  init.lower = 0, init.upper = 0, max.iter = 30, show.iter = FALSE)
```

Arguments

data	A vector of experimental data (non-negative integers as mutant count).
w	Relative fitness defined as (mutant growth rate)/(nonmutant growth rate).
alpha	The confidence coefficient is $1-\alpha$.
tol	Tolerance parameter to control for numerical accuracy.
init.m	Initial guess for the parameter m , often the default is sufficient.
init.lower	Initial guess for the lower limit of the confidence interval.
init.upper	Initial guess for the upper limit of the confidence interval.
max.iter	Maximum number of numerical iterations.
show.iter	To show the numerical iteration process.

Value

A pair of real numbers as the confidence limits.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

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

See Also

confint.LD

Examples

```
confint.MK(demerec.data, w=0.75)
```

`crane.data`*Experimental data of Crane, Thomas and Jones*

Description

A pair of fluctuation experiments conducted by Crane et al. (1996). For the first experiment, $N_t=3.6e9$, and the second $N_t=3.9e9$. In both experiments, a 0.2 ml portion was plated from each of the eleven 2.0-ml cultures. Thus, plating efficiency is $e=0.1$ in both experiments.

References

GJ Crane, SM Thomas, ME Jones, A modified Luria-Delbruck fluctuation assay for estimating and comparing mutation rates, *Mutation Research* 354 (1996) 171-182.

Examples

```
newton.LD.plating(unlist(crane.data[1]), 0.1)
```

`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)
```

ford.data	<i>Data of Mycobacterium tuberculosis Mutation by Fort et al.</i>
-----------	---

Description

One of several fluctuation experiments performed by Ford et al. (2011) to study Mycobacterium tuberculosis mutation rates. Nt=1.08e9.

References

CB Ford, PL Lin, MR Chase, RR Shah, O Iartchouk, J Galagan, N. Mohaideen, TR Ioerger, JC Sacchettini, M Lipsitch, JL Flynn, SM Fortune, Use of whole genome sequencing to estimate the mutation rate of Mycobacterium tuberculosis during latent infection, Nature Genetics 43 (2011) 482-486. See supplementary information.

Examples

```
newton.LD(ford.data)
confint.LD(ford.data) / (1.08e9)
```

import.excel.data	<i>Importing Data from an Excel File</i>
-------------------	--

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	<i>Importing Data from a Text File</i>
------------------	--

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 from the text file.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

LD.p0.est

The Classic P0 Method of Luria and Delbruck

Description

This is one of the two methods proposed in the classic paper of Luria and Delbruck (1943) to estimate the expected number of mutations per test tube, now commonly known as m . An important property of this method is often overlooked: the P0 method does not rely on the assumption that mutant cells and wild type cells grow at the same rate.

Usage

```
LD.p0.est(data)
```

Arguments

data a vector of non-negative integers representing mutant cell counts.

Value

An estimate of m , the expected number of mutations.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

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

See Also

newton.LD

Examples

```
LD.p0.est(cairns.foster.data)
```

likely.average

A family of Estimators of m Based on Delbruck's Concept of the Likely Average

Description

The concept of the likely average (Luria and Delbruck 1943) spawns a family of estimators. The well-known method of Lea and Coulson (1949) is a member of that family. However, most of these estimators are given solely for their pedagogical values and historical interest.

Usage

```
likely.average(data, b = 1.24, classic = FALSE, init.m = 0, tol = 1e-09,  
              max.iter = 25, use.median = TRUE, show.iter = FALSE)
```

Arguments

data	A vector of experimental data.
b	Setting b=1.24 (default) gives the Lea-Coulson estimator when the variable classic is set to False.
classic	If classic is true, the estimator depends on the number of cultures in an experiment. That was Delbruck's original idea.
init.m	An initial guess of m.
tol	Tolerance limit to control accuracy.
max.iter	Maximum number of iterations in the Newton procedure.
use.median	Either the sample median (default) or the sample mean is used in the equation.
show.iter	Show iteration history.

Value

A real number as an estimate of m.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

- E. A. Lea, C. A. Coulson, The distribution of the numbers of mutants in bacterial populations. *J Genetics* 49 (1949) 264-285.
- S. E. Luria, M. Delbruck, Mutations of bacteria from virus sensitivity to virus resistance. *Genetics* 28 (1943) 491-511.

See Also

newton.LD

Examples

```
likely.average(demerec.data)
```

```
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 $\phi=1-N_0/N_t$.

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 regarded 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

LRT.MK

*Likelihood Ratio Test under the Mandelbrot-Koch Model***Description**

The function computes the likelihood ratio test statistics. The algorithm is a generalization of a simpler algorithm given in Zheng (2015).

Usage

```
LRT.MK(X1, X2, R = 1, w1 = 1, w2 = 1, init.mc = 0, init.m1 = 0,
       init.m2 = 0, tol = 1e-09, show.iter = FALSE)
```

Arguments

X1	Data from the first experiment.
X2	Data from the second experiment.
R	Ratio of final cell population sizes, $R=(N_t \text{ of second experiment})/(N_t \text{ of first experiment})$.
w1	Relative fitness in the first experiment.
w2	Relative experiment in the second experiment.
init.mc	Initial guess of m under the null hypothesis.
init.m1	Initial guess for m1, m of the first experiment.
init.m2	Initial guess for m2, m of the second experiment.
tol	Tolerance parameter to control numerical accuracy.
show.iter	Logical variable to allow for display of the numerical iteration process.

Value

A pair of real numbers: (test statistic, p-value).

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, Methods for comparing mutation rates using fluctuation assay data, *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis* 777 (2015) 20-22.

Examples

```
x=simu.cultures(20,1e-8,1,1.2,50,2e8)
y=simu.cultures(20,1e-8,1,1.4,50,3e8)
LRT.MK(x,y,R=1.5,w1=1.2,w2=1.4)
```

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)
```

newcombe.data

Classic Experimental Data of Howard B. Newcombe

Description

The data frame contains 8 fluctuation experiments, labeled exptA, exptB, ... , exptH. Each experiment consists of 25 cultures (0.2ml). Initial numbers of cells (N_0) are: (10,10,10,10,1e4,1e4,1e4,1e4). Terminal cell population sizes (N_t) are: (3.1e8, 4.6e8, 2.5e8, 2.8e8, 4.2e8, 3.7e8, 3.2e8, 3.8e8).

References

HB Newcombe (1948) Delayed phenotypic expression of spontaneous mutations in *Escherichia Coli*, *Genetics* 33:447-476

Examples

```
LD.p0.est(unlist(newcombe.data[1]))
newton.LD(unlist(newcombe.data[1]))
```

newton.Haldane

Computing MLE of the Mutation Rate Under the Haldane Model

Description

This function computes maximum likelihood estimates of μ , 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 μ 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

<code>data</code>	a data vector, containing numbers of mutant cells in all tubes.
<code>g</code>	the number of generations.
<code>N0</code>	the number of initial cells.
<code>tol</code>	tolerance parameter to control convergence.
<code>init.mu</code>	an initial guess of μ can be supplied.
<code>max.iter</code>	maximum number of iterations allowed.
<code>show.iter</code>	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

<code>data</code>	a data vector, containing numbers of mutant cells in all test tubes.
<code>phi</code>	this parameter is defined by $1-N_0/N_t$, usually <code>phi</code> is taken as unity.
<code>tol</code>	tolerance parameter to control convergence.
<code>init.m</code>	an initial guess of m , it is rarely needed.
<code>max.iter</code>	maximum number of iterations.
<code>show.iter</code>	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	<i>Computing Maximum Likelihood Estimate of m that Adjusts for Plating Efficiency</i>
-------------------	--

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

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

newton.MK

*Computing MLE of m Under the Mandelbrot-Koch Model***Description**

This function computes maximum likelihood estimate of m , the expected number of mutations per culture, under the Mandelbrot-Koch model that allows for differential growth between mutants and nonmutants (Mandelbrot 1974; Koch 1982). The relative fitness (w) is usually obtained from a fitness assay (aka competition assay). The algorithm is a simple specialization of a more general algorithm given in Zheng (2005).

Usage

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

Arguments

<code>data</code>	A data vector containing the numbers of mutant cells from a fluctuation experiment.
<code>w</code>	Relative fitness define as (growth rate of mutants)/(growth rate of nonmutnats).
<code>tol</code>	Tolerance parameter to set a desired convergence criterion.
<code>init.m</code>	An initial guess of m , it is rarely needed.
<code>max.iter</code>	A parameter to set the maximum number of iteration for the Newton-Raphson procedure.
<code>show.iter</code>	A logical variable to show the numerical iteration process.

Value

A positive number as an MLE of m .

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

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

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 (1974) 437-444.

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

See Also

`newton.LD`

Examples

```
newton.MK(demerec.data, w=1.25)
```

 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 $N_0=1$ 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 μ . 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 μ .
mu.low	plot.likelihood.Haldane plots the log-likelihood function for μ 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.

<code>lik.col</code>	color for the log-likelihood function.
<code>mle.col</code>	color for the vertical bar that marks the maximum likelihood estimate of m.
<code>title</code>	title to be given to the graph.
<code>x.lab</code>	x label.
<code>y.lab</code>	y label.
<code>show.secant</code>	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, init.m=0, 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

<code>data</code>	A vector of experimental data (non-negative integers).
<code>init.m</code>	An initial guess of m for computing MLE of m.
<code>m.low</code>	plot.likelihood.LD plots the log-likelihood function for m ranging from between m.low to m.up.
<code>m.up</code>	see above.
<code>plot.pts</code>	number of points used to plot the log-likelihood functions. A larger number of points yields a smoother graph, but takes more computing time.
<code>lik.col</code>	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)
```

```
plot.likelihood.MK Plotting the Log-Likelihood Function under the Mandelbrot-Koch Model
```

Description

This function plots the log-likelihood function under the Mandelbrot-Kock mutation model. A value for the relative fitness parameter w is often obtained via a fitness assay. The algorithm for computing the maximum likelihood estimate of m is a specialization of an algorithm given in Zheng (2005).

Usage

```
plot.likelihood.MK(data, w = 1, m.low = -1, m.up = -1, init.m = 0,
  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.
w	Relative fitness.
m.low	The range of m for the plot is from m_{low} to m_{up} .
m.up	See above.
init.m	An initial guess of the parameter m .
plot.pts	Number of points used to plot the log-likelihood function.
lik.col	Color for the log-likelihood function.
mle.col	Color of the vertical line segment indicating the maximum likelihood estimate of m .
title	Plot title.
x.lab	Plot x-label.
y.lab	Plot y-label.
show.secant	A secant line will be drawn at the maximum of the log-likelihood function.

Value

None. A plot is created.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

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

Examples

```
plot.likelihood.MK(demerec.data, w=1.2)
```

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 μ 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 containing the first $n+1$ 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 ϕ , 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 ϕ was not considered at the time.

Usage

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

Arguments

m	expected number of mutations.
ϕ	This parameter is defined as $1-N_0/N_t$.
k	It computes the first $k+1$ probabilities, starting with the zeroth probability.

Details

Most implementations did not include the clock parameter ϕ .

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	<i>Calculating the Luria-Delbruck Mutant Cell Distribution that Adjusts for Plating Efficiency</i>
-----------------	--

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)
```

 prob.MK

Calculating the Mandelbrot-Koch Probability Distribution

Description

This mutant distribution was first proposed by Mandelbrot (1974). Kock (1982) and Stewart et al. (1990) made independent contributions. Explicit algorithm for computing the probabilities were given by Zheng (2005).

Usage

```
prob.MK(m, w = 1, n)
```

Arguments

m	Expected number of mutation per culture.
w	Relative fitness $w = (\text{birth rate of mutants}) / (\text{birth rate of nonmutants})$.
n	The first $n+1$ probabilities will be computed.

Value

A vector of $n+1$ probabilities.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

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

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 (1974) 437-444.

F. M. Stewart, D. M. Gordon, B. R. Levin, Fluctuation analysis: the probability distribution of the number of mutants under different conditions, *Genetics* 124 (1990) 175-185.

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

See Also

prob.LD, prob.Haldane

Examples

```
prob.MK(2.3, 1.5, 20)
```

simu.Bartlett

Simulating a Fluctuation Experiment Under the Bartlett Model

Description

This function simulates the dynamics of (X_t, Y_t) , where X_t is the number of nonmutant cells at time t , and where Y_t is the number of mutant cells at time t . This model was first proposed by Bartlett (1955) and further studied by Zheng (2008). The simulation algorithm is similar to that given on page 281 of Renshaw (1991).

Usage

```
simu.Bartlett(b1, b2, mu, N0, T, max.events = 1e+10, show.growth = FALSE)
```

Arguments

b1	Wild cell growth rate
b2	Mutant cell growth rate
mu	The mutation parameter "mu", giving a mutation rate of $\alpha = \mu / (b1 + \mu)$.
N0	Initial number of wild cells.
T	Terminating time.
max.events	Maximum number of simulation iterations; if exceeded, return value will be NA.
show.growth	A logical variable; if set to TRUE, cellular dynamics will be shown.

Value

A pair of non-negative integers in the form of (wild, mutant).

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

M.S. Bartlett, An Introduction to Stochastic Processes, Cambridge University Press, (1955).
 E. Renshaw, Modeling Biological Populations in Space and Time, Cambridge University Press (1991).
 Q. Zheng, On Bartlett's formulation of the Luria-Delbruck mutation model, Mathematical Bio-sciences 215:48-54 (2008).

See Also

simu.cultures, simu.Haldane

Examples

```
simu.Bartlett(1, 1, 5e-3, 1, 5.8, show.growth=TRUE)
```

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, b_1 , can differ from that of mutant cells, b_2 . The expected number of mutations is $m = \mu \cdot (N_t - N_0) / b_1$. The simulation algorithm is described in Zheng (2002).

Usage

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

Arguments

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

<code>gen</code>	The number of generations of cell division.
<code>mu</code>	The mutation rate
<code>culture</code>	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	<i>Simulating the Numbers of Mutant Cells for the Last Two Generations under the Haldane Model</i>
---------------	--

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	<i>Simulating a Fluctuation Experiment Under a variation of the Haldane Model</i>
-------------	---

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	<i>Simulating the Numbers of Mutant Cells for the Last Two Generations Under the Kimmel Model</i>
--------------	---

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)`

`wh.data`*Experimental Data of Werngren and Hoffner*

Description

There are 13 fluctuation experiments in this data object. From each 5ml culture a sample of 1ml was plated, and hence the plating efficiency was $e=0.2$. Terminal cell population sizes are $c(2.3, 0.5, 1.3, 1.0, 1.6, 1.0, 1.0, 0.8, 1.2, 0.6, 0.8, 0.9, 0.9)$ times $1e8$ per ml. Thus, for example, $N_t=1.15e9$ for the first experiment.

References

J Werngren, SE Hoffner, Drug-susceptible Mycobacterium tuberculosis Beijing genotype does not develop mutation-conferred resistance to rifampin at an elevated rate, Journal of clinical microbiology 41 (2003) 1520-1524.

Examples`newton.LD.plating(unlist(wh.data[1]), e=0.2)`

Index

*Topic \textasciitilde\dekwd1

compare.LD, 3
LD.p0.est, 11
likely.average, 11
LRT.MK, 15
prob.MK, 28
simu.Bartlett, 29
simu.Haldane2, 32
simu.Kimmel2, 33

*Topic \textasciitilde\dekwd2

compare.LD, 3
LD.p0.est, 11
likely.average, 11
LRT.MK, 15
prob.MK, 28
simu.Bartlett, 29
simu.Haldane2, 32
simu.Kimmel2, 33

*Topic datasets

boshoff.data, 2
crane.data, 8
ford.data, 9
newcombe.data, 16
wh.data, 34

*Topic package

rsalvador-package, 1

boshoff.data, 2

cairns.foster.data, 2
compare.LD, 3
confint.Haldane, 4
confint.LD, 5
confint.LD.plating, 6
confint.MK, 7
crane.data, 8

demerec.data, 8

export.text.data, 8

ford.data, 9

import.excel.data, 9
import.text.data, 10

LD.p0.est, 11
likely.average, 11
log.likelihood.Haldane, 13
log.likelihood.LD, 13
log.likelihood.LD.plating, 14
LRT.MK, 15
luria.16.data, 16

newcombe.data, 16
newton.Haldane, 17
newton.LD, 18
newton.LD.plating, 19
newton.MK, 20
niccum.data, 21

plot.likelihood.Haldane, 21
plot.likelihood.LD, 22
plot.likelihood.LD.plating, 23
plot.likelihood.MK, 24
prob.Haldane, 25
prob.LD, 26
prob.LD.plating, 27
prob.MK, 28

rsalvador (*rsalvador-package*), 1
rsalvador-package, 1

simu.Bartlett, 29
simu.cultures, 30
simu.Haldane, 31
simu.Haldane2, 32
simu.Kimmel, 32
simu.Kimmel2, 33

wh.data, 34