Package 'rsalvador'

November 26, 2020

Version 1.8

Date 2020-11-26

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

Author Qi Zheng <qzheng@tamu.edu>

Maintainer Qi Zheng <qzheng@tamu.edu>

Depends R (>= 2.15.0), hypergeo, gdata

Description Like its predecessor SALVADOR, rSalvador is a software tool intended for experimentalists running fluctuation experiments in the laboratory, and for biology educators and students teaching or learning fluctuation assay in the classroom. rSalvador comprises 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, variation in final cell population size or plating efficiency. rSalvador can also be used to compare mutation rates using likelihood ratio tests and to construct confidence intervals for ratios of mutation rates in different experiments. In addition to the usual continuous-time mutation model, rSalvador can accommodate Haldane's discrete-time mutation model. Furthermore, rSalvador can compute mutant cell distribution functions, plot log-likelihood functions, simulate fluctuation experiments and determine sample size.

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). An overview of the package's functionalities is given in Zheng (2015).

Details

Package:	rSalvador
Type:	Package
Version:	1.8
Date:	2020-11-26
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.

Q. Zheng, A new practical guide to the Luria-Delbruck protocol, Mutation Research/Fundamental and Molecular Mechanisms of Metagenesis 781 (2015) 7-13.

`bayes.foldchange.LD.plating`
A Bayesian Approach to Mutation Rate Fold Change Under an LD model with Imperfect Plating Efficiency

---

Description

The function considers two fluctuation experiments. The mutation rate in one experiment is regarded as a baseline rate, and the second mutation rate is expressed as mutation rate fold change relative to the baseline rate. Normal prior distributions are assumed for both parameters: the baseline rate and the fold change. Posterior distributions of the logarithms of the two parameters are then simulated using an MCMC method.

Usage

```
bayes.foldchange.LD.plating(y1, y2, N1, N2, e1 = 0.1, e2 = 0.1,
       s0 = 0.4, s1 = 0.5, init.base = 1e-08, init.fold = 1.6,
       v0 = 100, v1 = 100, iter = 1100, burn = 100,
      thin = 1, alpha = 0.05, short.out = TRUE, show.simu = FALSE)
```

Arguments

y1	Mutant count data for the first experiment.
y2	Mutant count data for the second experiment.
N1	Average final cell population size of the first experiment.
N2	Average final cell population size of the second experiment.
e1	Plating efficiency in the first experiment, must be smaller than 1.0.
e2	Plating efficiency in the second experiment, must be smaller than 1.0.
s0	Tuning parameter controlling the acceptance rate for the logarithm of the baseline rate, which is the variance of a normal proposal distribution.
s1	Tuning parameter controlling the acceptance rate for the logarithm of the fold change, which is the variance of a normal proposal distribution.
init.base	An initial guess of the baseline mutation rate.
init.fold	An initial guess of the mutation rate fold change.
v0	Prior variance for the logarithm of the baseline mutation rate.
v1	Prior variance for the logarithm of the mutation rate fold change.
iter	Total number of MCMC iterations.
burn	The burn-in period used in preprocessing the MCMC chains.
thin	Thinning parameter used in preprocessing the MCMC chains.
alpha	The Bayesian confidence interval will have an approximate confidence coefficient of 1-alpha.
short.out	If short.out=TRUE, the function gives only a summary of the fold change posterior distribution. Otherwise, it gives both Markov chains as well as summary statistics.
show.simu	A logic variable controlling the display of the MCMC simulation process.

Value

If short.out=TRUE, the function gives the median and a credible confidence interval for the fold change. Otherwise, the output is of the form list(accept.rate, list(stats1, stats2), list(chain1, chain2)), where accept.rate contains the two acceptance rate, where stats1 and stats2 are the two summaries of the two chains, and where chain1 and chain2 are the two preprocessed Markov chains

Author(s)

Qi Zheng <qzheng@tamu.edu>

Examples

```
y1 = wh.data[[1]]; y2 = wh.data[[2]]; N1 = 2.3e8; N2 = 0.5e8;
bayes.foldchange.LD.plating(y1, y2, N1, N2, e1 = 0.4, e2 = 0.4)
```

---

bayes.foldchange.MK

A Bayesian Approach to Mutation Rate Fold Change

Description

The function considers two fluctuation experiments. The mutation rate in one experiment is regarded as a baseline rate, and the second mutation rate is expressed as mutation rate fold change relative to the baseline rate. Normal prior distributions are assumed for both parameters: the baseline rate and the fold change. Posterior distributions of the logarithms of the two parameters are then simulated using an MCMC method.

Usage

```
bayes.foldchange.MK(y1, y2, N1, N2, w1 = 1, w2 = 1,
      s0 = 0.4, s1 = 0.5, init.base = 1e-08, init.fold = 1.6,
      v0 = 100, v1 = 100, iter = 1100, burn = 100, thin = 1,
      alpha = 0.05, short.out = TRUE, show.simu = FALSE)
```

Arguments

y1	Mutant count data for the first experiment.
y2	Mutant count data for the second experiment.
N1	Average final cell population size of the first experiment.
N2	Average final cell population size of the second experiment.
w1	Mutant relative fitness in the first experiment.
w2	Mutant relative fitness in the second experiment.
s0	Tuning parameter controlling the acceptance rate for the logarithm of the baseline rate, which is the variance of a normal proposal distribution.
s1	Tuning parameter controlling the acceptance rate for the logarithm of the fold change, which is the variance of a normal proposal distribution.
init.base	An initial guess of the baseline mutation rate.
init.fold	An initial guess of the mutation rate fold change.
v0	Prior variance for the logarithm of the baseline mutation rate.
v1	Prior variance for the logarithm of the mutation rate fold change.
iter	Total number of MCMC iterations.
burn	The burn-in period used in preprocessing the MCMC chains.
thin	Thinning parameter used in preprocessing the MCMC chains.

alpha	The Bayesian confidence interval will have an approximate confidence coefficient of 1-alpha.
short.out	If short.out=TRUE, the function gives only a summary of the fold change posterior distribution. Otherwise, it gives both Markov chains as well as summary statistics.
show.simu	A logic variable controlling the display of the MCMC simulation process.

Value

If short.out=TRUE, the function gives the median and a credible confidence interval for the fold change. Otherwise, the output is of the form list(accept.rate, list(stats1, stats2), list(chain1, chain2)), where accept.rate contains the two acceptance rate, where stats1 and stats2 are the two summaries of the two chains, and where chain1 and chain2 are the two preprocessed Markov chains

Author(s)

Qi Zheng <qzheng@tamu.edu>

Examples

```
y1 = cairns.foster.data; y2 = demerec.data; N1 = 1.5e8; N2 = 1.9e8;
bayes.foldchange.MK(y1, y2, N1, N2, init.base = 2.5e-8,
              init.fold = 2.5, burn = 1, iter = 5)
```

---

boot.foldchange.LD.plating

A Bootstrap Approach to Mutation Rate Fold Change Under an LD Model with Imperfect Plating Efficiency

---

Description

A bootstrapping sampling approach is utilized to construct a confidence interval for mutation rate fold change under an LD mutation model in which the plating efficiency is imperfect. The mutation rate fold change is defined as the mutation r ate in the second experiment divided by the mutation rate in the second experiment. A similar approach was first taken by Russell and March (2011)

Usage

```
boot.foldchange.LD.plating(y1, y2, N1, N2, e1 = 0.1, e2 = 0.1,
         alpha = 0.05, init.m1 = 0, init.m2 = 0, na.ok = FALSE, nboot = 10)
```

Arguments

y1	Mutant count data from the first experiment.
y2	Mutant count data from the second experiment.
N1	Average final cell population size of the first experiment.
N2	Average final cell population size of the second experiment.
e1	Plating efficiency in the first experiment, must be smaller than 1.0.
e2	Plating efficiency in the second experiment, must be smaller than 1.0.

alpha	The confidence interval will have an approximate confidence coefficient of 1-alpha.
init.m1	An initial guess of the expected number of mutations per culture in the first experiment.
init.m2	An initial guess of the expected number of mutations per culture in the second experiment.
na.ok	Not all bootstrap samples will allow the computational process for mutation rates to converge, often due to poorly chosen starting values. If na.ok is set to TRUE, such recalcitrant samples will be discarded before calculating a Bayesian confident interval. Otherwise, it stops further data processing and simply returns an "NA" as output.
nboot	Number of bootstrap samples to be drawn.

Value

A vector of the form (lower confidence limit, median, upper confidence limit)

Author(s)

Qi Zheng <qzheng@tamu.edu>

References

M.S. Russell, J. C. March, Bootstrap estimation of confidence intervals on mutation rate ratios. Environmental and Molecular Mutagenesis 52 (2011) 385-396.

See Also

boot.foldchange.MK

Examples

```
y1 = wh.data[[1]]; y2 = wh.data[[2]]; N1 = 2.3e8; N2 = 0.5e8;
boot.foldchange.LD.plating(y1, y2, N1, N2, e1 = 0.4, e2 = 0.4)
```

---

boot.foldchange.MK *A Bootstrap Approach to Mutation Rate Fold Change*

Description

A bootstrapping sampling approach is utilized to construct a confidence interval for mutation rate fold change under an MK mutation model. The mutation rate fold change is defined as the mutation rate in the second experiment divided by the mutation rate in the second experiment. A similar approach was first taken by Russell and March (2011)

Usage

```
boot.foldchange.MK(y1, y2, N1, N2, w1 = 1, w2 = 1,
                   alpha = 0.05, init.m1 = 0, init.m2 = 0,
                   na.ok = FALSE, nboot = 10)
```

Arguments

y1	Mutant count data from the first experiment.
y2	Mutant count data from the second experiment.
N1	Average final cell population size of the first experiment.
N2	Average final cell population size of the second experiment.
w1	Mutant relative fitness in the first experiment.
w2	Mutant relative fitness in the second experiment.
alpha	The confidence interval will have an approximate confidence coefficient of 1-alpha.
init.m1	An initial guess of the expected number of mutations per culture in the first experiment.
init.m2	An initial guess of the expected number of mutations per culture in the second experiment.
na.ok	Not all bootstrap samples will allow the computational process for mutation rates to converge, often due to poorly chosen starting values. If na.ok is set to TRUE, such recalcitrant samples will be discarded before calculating a Bayesian confident interval. Otherwise, it stops further data processing and simply returns an "NA" as output.
nboot	Number of bootstrap samples to be drawn.

Value

A vector of the form (lower confidence limit, median, upper confidence limit)

Author(s)

Qi Zheng <qzheng@tamu.edu>

References

M.S. Russell, J. C. March, Bootstrap estimation of confidence intervals on mutation rate ratios. Environmental and Molecular Mutagenesis 52 (2011) 385-396.

See Also

bayes.foldchange.MK

Examples

```
y1 = cairns.foster.data; y2 = demerec.data; N1 = 1.5e8; N2 = 1.9e8;
boot.foldchange.MK(y1, y2, N1, N2, nboot = 5)
```

boshoff.data *Experimental Data by Boshoff et al.*

Description

Data from 6 fluctuation experiments by Boshoff et al. (2003). The final cell population sizes Nt 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(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. Note Nt=1.5e8.

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, show.iter=FALSE)
```

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.
`show.iter`	a logic variable controlling the display of the iterative numerical process.

Value

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

Author(s)

Qi Zheng <qzheng@.tamu.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.B0` *Likelihood ratio-based confidence interval for m, under the Luria-Delbruck model, accounting for variation in Nt*

Description

This function computes a likelihood ratio-based confidence interval for m, taking into consideration of the variation in Nt, the cell population size immediately prior to plating. The algorithm is given in Zheng (2016), which is adapted from an algorithm given in Zheng (2011).

Usage

```
confint.B0(data, cv = 0.1, alpha = 0.05, tol = 1e-08, init.m = 5.1,
   init.lower = 0, init.upper = 0, max.iter = 30, show.iter = FALSE)
```

Arguments

`data`	A data vector representing the numbers of mutants in the test tubes.
`cv`	Coefficient of variation indicating the variation in Nt.
`alpha`	Significance level.
`tol`	Tolerance parameter to control numerical accuracy.
`init.m`	Initial guess for the parameter m.
`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 representing a confidence interval for m.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, A Bayesian two-level model for fluctuation assay, Genetica 139 (2011) 1409-1416.

Q. Zheng, A second look at the final number of cells in a fluctuation experiment, Journal of Theoretical Biology 401 (2016) 54-61.

See Also

newton.B0,confint.LD, confint.MK, confint.LD.plating

Examples

```
confint.B0(demerec.data,cv=0.1)
```

`confint.foldchange.LD` *Computing a confidence interval for mutation rate fold change*

Description

The algorithm regards the mutation rate in the first experiment as a baseline. The mutation rate fold change is estimated by the maximum likelihood method. This function then uses the profile likelihood method to compute confidence limits for the mutation rate fold change. This approach is similar to the one described in Zheng (2005).

Usage

```
confint.foldchange.LD(x, y, Nx, Ny, phi.x = 1, phi.y = 1, alpha = 0.05,
   init.base = 1e-08, init.fold = 1.6, init.low.base = -9,
   init.up.base = -9, init.low.fold = -9, init.up.fold = -9,
   max.iter = 30, tol = 1e-09, show.iter = FALSE)
```

Arguments

`x`	Mutant count data from the first experiment.
`y`	Mutant count data from the second experiment.
`Nx`	Final cell number Nt for the first experiment.
`Ny`	Final cell number Nt for the second experiment.
`phi.x`	The phi parameter in the mutant distribution for the first experiment, it is almost always 1.0 in practice.
`phi.y`	The phi parameter in the mutant distribution for the second experiment, it is almost always 1.0 in practice.
`alpha`	The confidence interval will have an approximate confidence coefficient of 1-alpha.
`init.base`	An initial guess of the baseline mutation rate, that is, the mutation rate in the first experiment.
`init.fold`	An initial guess of the mutation rate fold change.
`init.low.base`	An initial guess of the lower confidence limit for the baseline mutation rate.
`init.up.base`	An initial guess of the upper confidence limit for the baseline mutation rate.
`init.low.fold`	An initial guess of the left confidence limit for the mutation rate fold change.
`init.up.fold`	An initial guess of the right confidence limit for the mutation rate fold change.
`max.iter`	Maximum number of iterations allowed in the numerical iteration process.
`tol`	Tolerance controlling the numerical iteration process.
`show.iter`	A logic variable controlling the display of the numerical iteration process.

Value

A vector of the form (lower confidence limit, fold change, upper confidence limit)

Author(s)

Qi Zheng <qzheng@tamu.edu>

References

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

See Also

foldchange.LD

Examples

```
y1 = cairns.foster.data; y2 = demerec.data; N1 = 1.5e8; N2 = 1.9e8;
confint.foldchange.LD(y1, y2, N1, N2, init.base = 2.5e-8,
                      init.fold = 2.5,show.iter = TRUE)
```

---

confint.foldchange.LD.plating

CI for Mutation Rate Fold Change Under a Luria-Delbruck Model with Imperfect Plating Efficiency

---

Description

The algorithm regards the mutation rate in the first experiment as a baseline. The mutation rate fold change is estimated by the maximum likelihood method. This function then uses the profile likelihood method to compute confidence limits for the mutation rate fold change. This approach is similar to the one described in Zheng (2005).

Usage

```
confint.foldchange.LD.plating(x, y, Nx, Ny, e1 = 0.1, e2 = 0.1,
       alpha = 0.05, init.base = 1e-08, init.fold = 1.6,
       init.low.base = -9, init.up.base = -9, init.low.fold = -9,
       init.up.fold = -9, max.iter = 30, tol = 1e-09, show.iter = FALSE)
```

Arguments

x	Mutant count data from the first experiment.
y	Mutant count data from the second experiment.
Nx	Final cell number Nt in the first experiment.
Ny	Final cell number Nt in the second experiment.
e1	Plating efficiency in the first experiment.
e2	Plating efficiency in the second experiment.
alpha	The confidence interval for mutation rate foldl change will have an approximate confidence coefficient of 1-alpha.
init.base	An initial guess of the baseline mutation rate, that is, the mutation rate in the first experiment.

`init.fold`	An initial guess of the mutation rate fold change.
`init.low.base`	An initial guess of the lower confidence limit for the baseline mutation rate.
`init.up.base`	An initial guess of the upper confidence limit for the baseline mutation rate.
`init.low.fold`	An initial guess of the left confidence limit for the mutation rate fold change.
`init.up.fold`	An initial guess of the right confidence limit for the mutation rate fold change.
`max.iter`	Maximum number of iterations allowed in the numerical iteration process.
`tol`	Tolerance controlling the numerical iteration process.
`show.iter`	A logic variable controlling the display of the numerical iteration process.

Value

A vector of the form (lower confidence limit, fold change, upper confidence limit)

Note

newton.foldchange.LD.plating

Author(s)

Qi Zheng <qzheng@tamu.edu>

References

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

Examples

```
y1 = wh.data[[1]]; y2 = wh.data[[2]]; N1 = 2.3e8; N2 = 0.5e8;
confint.foldchange.LD.plating(y1, y2, N1, N2,
   e1 = 0.4, e2 = 0.4, init.base = 2e-8, show.iter = TRUE)
```

---

confint.foldchange.MK

Confidence Interval for Mutation Rate Fold Change Under an MK Model

---

Description

The algorithm regards the mutation rate in the first experiment as a baseline. The mutation rate fold change is estimated by the maximum likelihood method. This function then uses the profile likelihood method to compute confidence limits for the mutation rate fold change. This approach is similar to the one described in Zheng (2005).

Usage

```
confint.foldchange.MK(x, y, Nx, Ny, w1 = 1.0, w2 = 1.0, alpha = 0.05,
     init.base = 1e-08, init.fold = 1.6, init.low.base = -9,
     init.up.base = -9, init.low.fold = -9, init.up.fold = -9,
     max.iter = 30, tol = 1e-09, show.iter = FALSE)
```

Arguments

x	Mutant count data from the first experiment.
y	Mutant count data from the second experiment.
Nx	Final cell number Nt for the first experiment.
Ny	Final cell number Nt for the second experiment.
w1	Relative fitness of mutants in the first experiment.
w2	Relative fitness of mutants in the second experiment.
alpha	The confidence interval will have an approximate confidence coefficient of 1-alpha.
init.base	An initial guess of the baseline mutation rate, that is, the mutation rate in the first experiment.
init.fold	An initial guess of the mutation rate fold change.
init.low.base	An initial guess of the lower confidence limit for the baseline mutation rate.
init.up.base	An initial guess of the upper confidence limit for the baseline mutation rate.
init.low.fold	An initial guess of the left confidence limit for the mutation rate fold change.
init.up.fold	An initial guess of the right confidence limit for the mutation rate fold change.
max.iter	Maximum number of iterations allowed in the numerical iteration process.
tol	Tolerance controlling the numerical iteration process.
show.iter	A logic variable controlling the display of the numerical iteration process.

Value

A vector of the form (lower confidence limit, fold change, upper confidence limit)

Author(s)

Qi Zheng <qzheng@tamu.edu>

References

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

Examples

```
y1 = cairns.foster.data; y2 = demerec.data; N1 = 1.5e8; N2 = 1.9e8;
confint.foldchange.MK(y1, y2, N1, N2, w1 =0.95, w2 = 1.05,
     init.base = 2.5e-8, init.fold = 2.5, 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-9, 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-9, 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

`confint.MK` *Likelihood-Based Confidence Interval under the Mandelbrot-Koch Model*

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 counts).
`w`	Relative fitness defined as (mutant growth rate)/(nonmutant growth rate).
`alpha`	The confidence coefficient is 1-alpha.
`tol`	Tolerance parameter to control 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)
```

`confint.profile.m` *Profile Likelihood Confidence Intervals for m Under the Mandelbrot-Koch Model*

Description

This function computes profile likelihood confidence intervals for m under the Mandelbrot-Koch model (Mandelbrot 1974; Koch 1982). Here m is the expected number of mutations per culture. The Mandelbrot-Koch model allows for differential growth between mutants and nonmutants, that is, the relative fitness of mutants, w, can be different from 1.0. The adopted iterative algorithm is described in detail in Zheng (2005).

Usage

```
confint.profile.m(data, alpha = 0.05, init.low.m = -1, init.low.w = -1,
    init.up.m = -1, init.up.w = -1, init.m = -1, init.w = -1,
       max.iter = 30, tol = 1e-09, show.iter = FALSE)
```

Arguments

`data`	A vector containing mutant counts.
`alpha`	The confidence level is 1.0-alpha.
`init.low.m`	A starting value for the lower confidence limit of m.
`init.low.w`	A starting value for the lower confidence limit of w.
`init.up.m`	A starting value for the upper confidence limit of m.
`init.up.w`	A starting value for the upper confidence limit of w.
`init.m`	A starting value for the MLE of m, needed for computing the profile likelihood CI.
`init.w`	A starting value for the MLE of w, needed for computing the profile likelihood CI.
`max.iter`	Maximum number of numerical iterations.
`tol`	A tolerance parameter to control the number of iterations.
`show.iter`	A logical variable controlling the display of numerical iterations.

Value

A pair of real numbers representing the profile confidence interval for 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

confint.MK, confint.profile.w

Examples

```
confint.profile.m(demerec.data)
```

`confint.profile.w` *Profile Likelihood Confidence Intervals for w (Relative Fitness) Under the Mandelbrot-Koch Model*

Description

This function computes profile likelihood confidence intervals for w under the Mandelbrot-Koch model (Mandelbrot 1974; Koch 1982), which allows for differential growth between mutants and nonmutants. That is, the relative fitness of mutants, w, can be different from 1.0. The adopted iterative algorithm is described in detail in Zheng (2005).

Usage

```
confint.profile.w(data, alpha = 0.05, init.low.m = -1, init.low.w = -1,
    init.up.m = -1, init.up.w = -1, init.m = -1, init.w = -1,
       max.iter = 30, tol = 1e-09, show.iter = FALSE)
```

Arguments

`data`	A vector containing mutant counts.
`alpha`	The confidence level is 1.0-alpha.
`init.low.m`	A starting value for the lower confidence limit of m.
`init.low.w`	A starting value for the lower confidence limit of w.
`init.up.m`	A starting value for the upper confidence limit of m.
`init.up.w`	A starting value for the upper confidence limit of w.
`init.m`	A starting value for the MLE of m, needed for computing the profile likelihood CI.
`init.w`	A starting value for the MLE of w, needed for computing the profile likelihood CI.
`max.iter`	Maximum number of numerical iterations.
`tol`	A tolerance parameter to control the number of iterations.
`show.iter`	A logical variable controlling the display of numerical iterations.

Value

A pair of real numbers representing the profile confidence interval for 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

confint.MK, confint.profile.m

Examples

```
confint.profile.w(demerec.data)
```

`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, Nt=3.6e9, and the second Nt=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(crane.data[[1]],e=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. Nt=1.9e8.

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)
```

fisher.LD.plating *Computing Expected Fisher Information for m Under the Standard Luria-Delbruck Distribution with Partial Plating*

Description

This function computes approximate expected Fisher information for m under the standard Luria-Delbruck distribution with partial plating. The algorithm is adapted from that given in Zheng (2002). The expected Fisher information can be utilized in determining sample size (the number of cultures) in a fluctuation experiment.

Usage

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

Arguments

m	The excepted number of mutations.
e	The parameter of plating efficiency (0<e<1).
n	The number of terms kept in summing an infinite series that defines the expected Fisher information.

Value

A positive real number.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

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

See Also

fisher.MK

Examples

```
fisher.LD.plating(m=50.0,e=0.2,n=2000)
```

`fisher.MK` *Computing Expected Fisher Information for m Under the MK distribution*

Description

This function computes approximate expected Fisher information for m under the Mandelbrot-Koch distribution. The algorithm is adapted from that given in Zheng (2002). The expected Fisher information can be utilized in determining sample size (the number of cultures) in a fluctuation experiment.

Usage

```
fisher.MK(m, w, n)
```

Arguments

`m`	The expected number of mutations.
`w`	The parameter of relative fitness.
`n`	The number of terms kept in summing an infinite series that defines the expected Fisher information.

Value

A positive real number.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

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

See Also

fisher.LD.plating

Examples

```
fisher.MK(3.0,1.0,2000)
```

`ford.data` *Data of Mycobacterium tuberculosis Mutation by Fort et al.*

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 inforaamtion.

Examples

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

`foster.data` *Fluctuation Experimental Data by P. Foster*

Description

Data from an experiment conducted by P. Foster in 1994. The study was about a reversion of Lac- to Lac+ in the FC 40 strain of Escherichia coli. Three cultures were sampled to to estimate Nt. At the 1e-6 dilution, the three measurements were 702, 645 and 536. Therefore, Nt=6.16e8. The data first appeared in Zheng (2011).

References

Q. Zheng, A Bayesian two-level model for fluctuation assay, Genetica 139 (2011) 1409-1416.

Examples

```
newton.LD(foster.data)
```

`golden.benchmark.LD` *Maximum Likelihood Estimation of the Mutation Rate Accounting for Variation in Nt*

Description

Nt is the (non-mutant) cell population size immediately prior to plating. In a fluctuation experiment comprising n tubes, the final cell population sizes are Nt(1), Nt(2), ..., Nt(n). Measuring all Nt(i) is a daunting experimental challenge. However, theoretically, when all Nt(i) are known, a maximum likelihood estimate of m can be computed. Thus, this scenario serves as a theoretical benchmark model. This function uses the golden section search method to calculate an MLE of mu (the mutation rate), incorporating the measurements of Nt(i). The likelihood function is formed using the Luria-Delbruck distribution as described by Lea and Coulson (1949).

Usage

```
golden.benchmark.LD(data, Nt, mu.low = 1e-12, mu.up = 5e-07,
                tol = 1e-09, max.iter = 100, show.iter = FALSE)
```

Arguments

`data`	A vector containing the mutant counts.
`Nt`	A vector containing the final cell population sizes.
`mu.low`	A lower bound for the mutation rate.
`mu.up`	An upper bound for the mutation rate. Because exp(-m) will cause underflow when m exceeds 744, it is imperative to ensure that (max Nt(i))*mu.up < 700.
`tol`	A parameter to control numerical accuracy.
`max.iter`	Maximum number of iterations allowable in the golden section search process.
`show.iter`	To exhibit the step-by-step golden section search process.

Value

A real number as an estimate of the mutation rate, not the expected number of mutations (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.

Examples

```
golden.benchmark.LD(demerec.data,rep(1.9e8,30))
```

`golden.LD.B0` *Accounting for Varying Nt Under the Luria-Delbruck Model*

Description

This function uses the B0 distribution (Zheng 2010, 2011) to account for variation in Nt, the final cell population size prior to plating. The function employs the golden section search algorithm to find a value of m that maximizes the log likelihood functions.

Usage

```
golden.LD.B0(data, cv = 0.1, m.low = 0.1, m.up = 30, tol = 1e-8,
   max.iter = 60, show.iter = FALSE)
```

Arguments

`data`	A vector containing the mutant count data.
`cv`	Known (independently estimated) coefficient of variation for Nt, final cell population size.
`m.low`	A lower starting value of m for the golden section search.
`m.up`	An upper starting value of m for the golden section search.
`tol`	A tolerance parameter to control numerical precision.
`max.iter`	Maximum number of iterations.
`show.iter`	A logical variable controlling the exhibition of the numerical iteration process.

Value

A positive real number as an estimate of m, the expected number of mutations per culture.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, A new discrete distribution induced by the Luria-Delbruck mutation model, Statistics 44 (2010) 529-540.

Q. Zheng, A Bayesian two-level model for fluctuation assay, Genetica 139 (2011) 1409-1416.

See Also

newton.LD, newton.LD.plating

Examples

```
golden.LD.B0(demerec.data,cv=0.064)
```

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>

`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-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

LRT.LD *Likelihood Ratio Test under the Luria-Delbruck Model*

Description

The function computes the likelihood ratio test statistic to compare the mutation rates in two independent fluctuation experiments. Both experiments are modeled by the standard Luria-Delbruck distribution. The algorithm is described in Zheng (2016), which is a generalization of a simpler algorithm given in Zheng (2015).

Usage

```
LRT.LD(X1, X2, R = 1, phi1 = 1, phi2 = 1, init.mc = 0, init.m1 = 0,
     init.m2 = 0, tol = 1e-09, max.iter = 30, show.iter = FALSE)
```

Arguments

X1	Mutant count data from the first experiment.
X2	Mutant count data from the second experiment.
R	Ratio of final cell population sizes, R=(Nt of second experiment)/(Nt of first experiment).
phi1	"Time" parameter for the first experiment, usually set to unit.
phi2	"Time" parameter for the second experiment, usually set to unit.
init.mc	Initial guess of m under the null hypothesis – the combined model.
init.m1	Initial guess of m for the first experiment.
init.m2	Initial guess of m for the second experiment.
tol	Tolerance parameter to control numerical accuracy.
max.iter	Maximum number of iterations allowed in the numerical iteration process.
show.iter	Logical variable to control the 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.

Q. Zheng, Comparing mutation rates under the Luria-Delbruck protocol, Genetica 144 (2016) 351-359.

See Also

LRT.MK, LRT.LD.plating

Examples

```
x=simu.cultures(20,1e-8,1,1,50,2e8)
y=simu.cultures(20,1e-8,1,1,50,3e8)
LRT.LD(x,y,1.5)
```

`LRT.LD.plating` *Likelihood Ratio Test under the Classic Luria-Delbruck Model with Partial Plating*

Description

The function computes the likelihood ratio test statistics to compare the mutation rates in two independent fluctuation experiments. The algorithm is described in Zheng (2016), which is a generalization of a simpler algorithm given in Zheng (2015).

Usage

```
LRT.LD.plating(X1, X2, R = 1, e1 = 0.1, e2 = 0.1, init.mc = 0,
      init.m1 = 0, init.m2 = 0, tol = 1e-09, max.iter = 30, show.iter = FALSE)
```

Arguments

`X1`	Data from the first experiment.
`X2`	Data from the second experiment.
`R`	Ratio of final cell population sizes, R=(Nt of second experiment)/(Nt of first experiment).
`e1`	Plating efficiency in the first experiment.
`e2`	Plating efficiency 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.
`max.iter`	Maximum number of iterations allowed in the numerical iteration process.
`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@tamu.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.

Q. Zheng, Comparing mutation rates under the Luria-Delbruck protocol, Genetica 144 (2016) 351-359.

See Also

LRT.MK

Examples

```
R = 0.5 / 2.3
LRT.LD.plating(wh.data[[1]], wh.data[[2]], R = R, e1 = 0.4, e2 = 0.4)
```

`LRT.MK` *Likelihood Ratio Test under the Mandelbrot-Koch Model*

Description

The function computes the likelihood ratio test statistic that compares the mutation rates in two independent fluctuation experiments. The algorithm is described in Zheng (2016), which 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, max.iter = 30, show.iter = FALSE)
```

Arguments

`X1`	Data from the first experiment.
`X2`	Data from the second experiment.
`R`	Ratio of final cell population sizes, R=(Nt of second experiment)/(Nt 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.
`max.iter`	It controls the maximum number of iterations in the iterative numerical process.
`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.

Q. Zheng, Comparing mutation rates under the Luria-Delbruck protocol, Genetica 144 (2016) 351-359.

See Also

LRT.LD, LRT.LD.plating

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)
```

`mcmc.LD.plating` *Estimating mutation rates under the LD model (with partial plating) using an MCMC approach*

Description

This function assumes the classic Luria-Delbruck model under which relative fitness w=1. Plating is partial, and hence plating efficiency e<1.0. An MCMC method is used to simulate the posterior distribution of the logarithm of the mutation rate. A Gaussian prior distribution is assumed. For a general introduction to MCMC computing, see Albert (2007); for an application of MCMC technique in the analysis of fluctuation assay data, see Zheng (2011).

Usage

```
mcmc.LD.plating(y, Nt, e = 0.1, Iter = 100, init.mu = 2e-08,
          b0 = -15, v0 = 30, s0 = 0.8, show.simu = FALSE)
```

Arguments

`y`	A data vector containing the number of mutants in a fluctuation experiment.
`Nt`	Average final cell population size.
`e`	Plating efficiency.
`Iter`	Length of the simulated Markov chain.
`init.mu`	Initial guess of the mutation rate (not the expected number of mutations).
`b0`	Prior mean of log of the mutation rate.
`v0`	Variance in the Gaussian prior distribution.
`s0`	A tuning parameter to control acceptance rate; it is the variance of a Gaussian proposal distribution.
`show.simu`	A logical variable to control the display of the evolution of the MCMC process.

Value

list(acc.rate,chain) where acc.rate is the acceptance rate and where chain is the simulated Markov chain. Note that chain usually needs further processing, e.g., discarding a burn-in period and thinning, etc.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

J. Albert, Bayesian Computation Using R, Wiley, 2007.

Q. Zheng, A Bayesian approach for correcting for partial plating in fluctuation experiments, Genetics Research 93 (2011) 351-356.

See Also

mcmc.MK.test

Examples

```
mcmc.LD.plating(unlist(wh.data[1]),1.59e9,e=0.2,Iter=1000,s0=0.2)
```

---

mcmc.LD.plating.test

Comparison of mutation rates under the LD model (with partial plating) using an MCMC approach

---

Description

This function assumes the classic LD model, under which the relative fitness w=1. The classic approach to the comparison of mutation rates is the likelihood ratio test (LRT), which can be done with the LRT.MK functions, for example. The function allows for a Bayesian approach to the comparison of mutation rates. In the first experiment, the expected number of mutations per culture is N1*exp(b0), while in the second experiment that quantity is N2*exp(b0+b1). Therefore, the two mutation rates are exp(b0) and exp(b0+b1), respectively. Equality of the two mutation rates is equivalent to b1=0. Note that the ratio of the two mutation rates is mu2:mu1=exp(b1). For a general introduction to MCMC computing, see Albert (2007); for an application of MCMC technique in the analysis of fluctuation assay data, see Zheng (2011).

Usage

```
mcmc.LD.plating.test(x, y, N1, N2, e1 = 0.1, e2 = 0.1, Iter = 100,
    init.mu0 = -20, init.mu1 = 0.05, mu0 = -20.7, mu1 = 2.7,
    v0 = 30, v1 = 30, s0 = 0.5, s1 = 0.5, show.simu = FALSE)
```

Arguments

x	A data vector containing mutant counts in the first experiment.
y	A data vector containing mutant counts in the second experiment.
N1	Average final cell population size of the first experiment.
N2	Average final cell population size of the second experiment.
e1	Plating efficiency in the first experiment.
e2	Plating efficiency in the second experiment.
Iter	Number of MCMC simulation cycles.
init.mu0	Initial guess of the logarithm of the mutation rate in the first experiment.
init.mu1	Initial guess of the difference between log(second mutation rate) and log(first mutation rate).
mu0	Prior mean of log(first mutation rate)
mu1	Prior mean of log(second mutation rate) - log(first mutation rate).
v0	Prior variance of log(first mutation rate).
v1	Prior variance of mu1.
s0	Tuning parameter to control the acceptance rate of b0, the variance of a normal proposal distribution.
s1	Tuning parameter to control the acceptance rate of b1, the variance of a normal proposal distribution.
show.simu	A logic variable to control the display of the evolution of the MCMC simulation process.

Value

list(c(a1,a2), chain), where a1 and a2 are the two acceptance rates, and chain is a Iter by 2 matrix containing the simulated posterior distributions of mu0 and mu1.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

J. Albert, Bayesian Computation Using R, Wiley, 2007.

Q. Zheng, A Bayesian approach for correcting for partial plating in fluctuation experiments, Genetics Research 93 (2011) 351-356.

See Also

mcmc.MK.test

Examples

```
x=unlist(wh.data[1])
y=unlist(wh.data[2])
z=mcmc.LD.plating.test(x,y,1.15e9,2.5e8,e1=0.2,e2=0.2,Iter=100)
exp(median(z[[2]][,1]))
```

---

mcmc.MK — *Estimating mutation rates under the MK model using an MCMC approach*

Description

An MCMC method is used to simulate the posterior distribution of the logarithm of the mutation rate. A Gaussian prior distribution is assumed. For a general introduction to MCMC computing, see Albert (2007); for an application of MCMC technique in the analysis of fluctuation assay data, see Zheng (2011).

Usage

```
mcmc.MK(y, Nt, w = 1, Iter = 100, init.mu = 2e-08, b0 = -15,
    v0 = 30, s0 = 0.8, show.simu = FALSE)
```

Arguments

y	A data vector containing the number of mutants in a fluctuation experiment.
Nt	Average final cell population size.
w	Relative fitness defined as (division rate of mutants)/(division rate of wide-type cells).
Iter	Length of the simulated Markov chain.
init.mu	Initial guess of the mutation rate (not the expected number of mutations).
b0	Prior mean of log of the mutation rate.

v0	Variance in the Gaussian prior distribution.
s0	A tuning parameter to control acceptance rate; it is the variance of a Gaussian proposal distribution.
show.simu	A logical variable to control the display of the evolution of the MCMC process.

Value

list(acc.rate,chain) where acc.rate is the acceptance rate and where chain is the simulated Markov chain. Note that chain usually needs further processing, e.g., discarding a burn-in period and thinning, etc.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

J. Albert, Bayesian Computation Using R, Wiley, 2007.

Q. Zheng, A Bayesian approach for correcting for partial plating in fluctuation experiments, Genetics Research 93 (2011) 351-356.

See Also

mcmc.MK.test

Examples

```
mcmc.MK(demerec.data,1.9e8,w=1,Iter=100)
```

---

mcmc.MK.test *Comparison of mutation rates using an MCMC approach*

Description

The classic approach to the comparison of mutation rates is the likelihood ratio test (LRT), which can be done with the LRT.MK functions, for example. The function allows for a Bayesian approach to the comparison of mutation rates. In the first experiment, the expected number of mutations per culture is N1*exp(b0), while in the second experiment that quantity is N2*exp(b0+b1). Therefore, the two mutation rates are exp(b0) and exp(b0+b1), respectively. Equality of the two mutation rates is equivalent to b1=0. Note that the ratio of the two mutation rates is mu2:mu1=exp(b1). For a general introduction to MCMC computing, see Albert (2007); for an application of MCMC technique in the analysis of fluctuation assay data, see Zheng (2011).

Usage

```
mcmc.MK.test(x, y, N1, N2, w1 = 1, w2 = 1, Iter = 100, init.mu0 = -20,
    init.mu1 = 0.05, mu0 = -20.7, mu1 = 2.7, v0 = 30, v1 = 30, s0 = 0.5,
    s1 = 0.5, show.simu = FALSE)
```

Arguments

x	A data vector containing mutant counts in the first experiment.
y	A data vector containing mutant counts in the second experiment.
N1	Average final cell population size of the first experiment.
N2	Average final cell population size of the second experiment.
w1	Relative fitness in the first experiment.
w2	Relative fitness in the second experiment.
Iter	Number of MCMC simulation cycles.
init.mu0	Initial guess of the logarithm of the mutation rate in the first experiment.
init.mu1	Initial guess of the difference between log(second mutation rate) and log(first mutation rate).
mu0	Prior mean of log(first mutation rate)
mu1	Prior mean of log(second mutation rate) - log(first mutation rate).
v0	Prior variance of log(first mutation rate).
v1	Prior variance of mu1.
s0	Tuning parameter to control the acceptance rate of b0, the variance of a normal proposal distribution.
s1	Tuning parameter to control the acceptance rate of b1, the variance of a normal proposal distribution.
show.simu	A logic variable to control the display of the evolution of the MCMC simulation process.

Value

list(c(a1,a2), chain), where a1 and a2 are the two acceptance rates, and chain is a Iter by 2 matrix containing the simulated posterior distributions of mu0 and mu1.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

J. Albert, Bayesian Computation Using R, Wiley, 2007.

Q. Zheng, A Bayesian approach for correcting for partial plating in fluctuation experiments, Genetics Research 93 (2011) 351-356.

See Also

mcmc.MK

Examples

```
mcmc.MK.test(demerec.data,ford.data,1.9e8,1.08e9,1,1,100)
```

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 (N0) are: (10,10,10,10,1e4,1e4,1e4,1e4). Terminal cell population sizes (Nt) are: (3.1e8, 4.6e8, 2.5e8, 2.8e8, 4.2e8, 3.7e8, 3.2e8, 3.8e8).

References

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

Examples

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

newton.B0 — *Computing MLE of m under the Luria-Delbruck model, accounting for variation in Nt*

Description

This function computes the MLE of m by accounting for the variation in Nt, the cell population size immediately prior to plating. Let the final cell population sizes be Nt(1), Nt(2), ..., Nt(n). If the coefficient of variation (CV) of Nt(1), ..., Nt(n) is large, an MLE of m can be computed by taking into consideration of the CV for Nt. An underlying assumption is that each test tube has a different values of m, i.e., m(1), m(2), ..., m(n), which are generated by a Gamma distribution with mean m0 and coefficient of variation CV. The output of newton.B0 is an estimate of m0, computed using the Newton-Raphson algorithm. The function name reflects the fact that a gamma mixture of the Luria-Delbruck distribution is a B0 distribution (Zheng 2010). The specific algorithm is given in Zheng (2016), which is adapted from an algorithm given in Zheng (2011).

Usage

```
newton.B0(data, cv = 0.1, init.m = 0, tol = 1e-08, max.iter=30,
   show.iter = FALSE)
```

Arguments

data	A data vector containing numbers of mutant cells in all test tubes.
cv	Coefficient of variation, believed to represent the variability in Nt.
init.m	An initial guess of m.
tol	A tolerance parameter to control convergence.
max.iter	To control the maximum number of iterations during the Newton-Raphson process.
show.iter	To exhibit the intermediate values of m during the Newton-Raphson iteration process.

Value

A positive real number.

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, A new discrete distribution induced by the Luria-Delbruck mutation model, Statistics 44 (2010) 529-540.

Q. Zheng, A Bayesian two-level model for fluctuation assay, Genetica 139 (2011) 1409-1416.

Q. Zheng, A second look at the final number of cells in a fluctuation experiment, Journal of Theoretical Biology 401 (2016) 54-63.

See Also

newton.LD, newton.MK, newton.LD.plating

Examples

```
newton.B0(demerec.data,cv=0.1)
```

---

newton.foldchange.LD

Computing mutation rate fold change

Description

The algorithm regards the mutation rate in the first experiment as a baseline. The mutation rate fold change is estimated by the maximum likelihood method.

Usage

```
newton.foldchange.LD(x, y, Nx, Ny, phi.x = 1, phi.y = 1, tol = 1e-08, init.base
       init.fold = 1.6, max.iter = 30, no.log = TRUE, show.iter = FALSE)
```

Arguments

x	Mutant count data from the first experiment.
y	Mutant count data from the second experiment.
Nx	Final cell number Nt for the first experiment.
Ny	Final cell number Nt for the second experiment.
phi.x	The phi parameter in the mutant distribution for the first experiment, it is almost always 1.0 in practice.
phi.y	The phi parameter in the mutant distribution for the second experiment, it is almost always 1.0 in practice.
tol	Tolerance controlling the numerical iteration process.

init.base	An initial guess of the baseline mutation rate, that is, the mutation rate in the first experiment.
init.fold	An initial guess of the mutation rate fold change.
max.iter	Maximum number of iterations allowed in the numerical iteration process.
no.log	The algorithm computes the logarithm of a fold change, but it returns the fold change directly unless no.log is FALSE.
show.iter	A logic variable controlling the display of the numerical iteration process.

Value

A vector of the form (baseline mutation rate, fold change)

Author(s)

Qi Zheng <qzheng@tamu.edu>

See Also

confint.foldchange.LD

Examples

```
y1=cairns.foster.data; y2=demerec.data; N1=1.5e8; N2=1.9e8;
newton.foldchange.LD(y1,y2,N1,N2,init.base=2.5e-8,init.fold=2.5,show.iter=TRUE)
```

---

newton.foldchange.LD.plating

Mutation Rate Fold Change Under a Luria-Delbruck Model with Imperfect Plating Efficiency

---

Description

The algorithm regards the mutation rate in the first experiment as a baseline. The mutation rate fold change is estimated by the maximum likelihood method. This function then uses the profile likelihood method to compute confidence limits for the mutation rate fold change. This approach is similar to the one described in Zheng (2005).

Usage

```
newton.foldchange.LD.plating(x, y, Nx, Ny, e1 = 0.1, e2 = 0.1, tol = 1e-08, init
                             init.fold = 1.6, max.iter = 30, no.log = TRUE, show
```

Arguments

x	Mutant count data from the first experiment.
y	Mutant count data from the second experiment.
Nx	Final cell number Nt in the first experiment.
Ny	Final cell number Nt in the second experiment.
e1	Plating efficiency in the first experiment.

e2	Plating efficiency in the second experiment.
tol	Tolerance controlling the numerical iteration process.
init.base	An initial guess of the baseline mutation rate, that is, the mutation rate in the first experiment.
init.fold	An initial guess of the mutation rate fold change.
max.iter	Maximum number of iterations allowed in the numerical iteration process.
no.log	The algorithm computes the logarithm of a fold change, but it returns the fold change directly unless no.log is FALSE.
show.iter	A logic variable controlling the display of the numerical iteration process.

Value

A vector of the form (baseline mutation rate, fold change)

Author(s)

Qi Zheng <qzheng@tamu.edu>

References

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

See Also

confint.foldchange.LD.plating

Examples

```
y1 = wh.data[[1]]; y2 = wh.data[[2]]; N1 = 2.3e8; N2 = 0.5e8;
newton.foldchange.LD.plating(y1, y2, N1, N2, e1 = 0.4, e2 = 0.4, init.base = 2e-8, show.i
```

---

newton.foldchange.MK

Estimating Mutation Rate Fold Change Under an MK Model

Description

The algorithm regards the mutation rate in the first experiment as a baseline. The mutation rate fold change is estimated by the maximum likelihood method. This approach is similar to the one described in Zheng (2005).

Usage

```
newton.foldchange.MK(x, y, Nx, Ny, w1 = 1.0, w2 = 1.0, tol = 1e-09, init.base =
                     max.iter = 30, no.log = TRUE, show.iter = FALSE)
```

Arguments

x	Mutant count data from the first experiment.
y	Mutant count data from the second experiment.
Nx	Final cell number Nt in the first experiment.
Ny	Final cell number Nt in the second experiment.
w1	Relative fitness of mutants in the first experiment.
w2	Relative fitness of mutants in the second experiment.
tol	Tolerance controlling the numerical iteration process.
init.base	An initial guess of the baseline mutation rate, that is, the mutation rate in the first experiment.
init.fold	An initial guess of the mutation rate fold change.
max.iter	Maximum number of iterations allowed in the numerical iteration process.
no.log	The algorithm computes the logarithm of a fold change, but it returns the fold change directly unless no.log is FALSE.
show.iter	A logic variable controlling the display of the numerical iteration process.

Value

A vector of the form (baseline mutation rate, fold change)

Author(s)

Qi Zheng <qzheng@tamu.edu>

References

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

See Also

confint.foldchange.MK

Examples

```
y1 = cairns.foster.data; y2 = demerec.data; N1 = 1.5e8; N2 = 1.9e8;
newton.foldchange.MK(y1, y2, N1, N2, w1 =0.95, w2 = 1.05, init.base = 2.5e-8, init.fold =
```

`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.joint.MK` *Joint Estimation of m and w Under the Mandelbrot-Koch Model*

Description

This function computes maximum likelihood estimates of m and w simultaneously under the Mandelbrot-Koch model (Mandelbrot 1974; Koch 1982). Here m is the expected number of mutations per culture, and w is relative fitness of mutants. The Mandelbrot-Koch model allows for differential growth between mutants and nonmutants. The Newton-Raphson algorithm adopted is described in detail in Zheng (2005).

Usage

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

Arguments

`data`	A vector containing mutant counts.
`tol`	Tolerance parameter to control the number of numerical iterations.
`init.m`	A starting value for the m parameter, the expected number of mutations per culture.
`init.w`	A starting value for the w parameter, relative fitness of the mutants.
`max.iter`	Maximum number of Newton-Raphson iterations.
`show.iter`	A logical variable enabling the display of the numerical iteration process.

Value

A pair of non-negative numbers representing the maximum likelihood estimates of m and w.

Author(s)

Qi Zheng <qzheng@sph.tamhse.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.MK

Examples

```
newton.joint.MK(demerec.data)
```

`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-9, 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, New algorithms for Luria-Delbruck fluctuation analysis, Mathematical Biosciences 196 (2005) 198-214.

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-9, 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, A note on plating efficiency in fluctuation experiments, Mathematical Biosciences 216 (2008) 150-153.

See Also

newton.LD

Examples

```
newton.LD.plating(wh.data[[1]],e=0.4)
```

`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

`data`	A data vector containing the numbers of mutant cells from a fluctuation experiment.
`w`	Relative fitness define as (growth rate of mutants)/(growth rate of nonmutants).
`tol`	Tolerance parameter to set a desired convergence criterion.
`init.m`	An initial guess of m, it is rarely needed.
`max.iter`	A parameter to set the maximum number of iterations for the Newton-Raphson procedure.
`show.iter`	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 reported by 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 and the plating efficiency is 0.10.

References

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

Examples

```
newton.LD.plating(niccum.data,e=0.1)
```

p0.LD.plating *P0 method modified for partial plating*

Description

The classic P0 method proposed by Luria and Delbruck (1943) does not allow for partial plating. A modified P0 method that accounts for partial plating was known to Stewart (1991) and to Jones (1993). Algorithms for computing maximum likelihood estimates of m was later proposed by Zheng (2008).

Usage

```
p0.LD.plating(data, e)
```

Arguments

data	A data vector of mutant counts
e	A known plating efficiency

Value

An estimate of m, the expected number of mutations per culture

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

M.E. Jones, Accounting for plating efficiency when estimating spontaneous mutation rates, Mutation Research 292 (1993) 187-189.

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

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

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

See Also

newton.LD.plating

Examples

```
p0.LD.plating(luria.16.data,e=0.4)
```

---

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, 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

data	A vector of experimental data (non-negative integers).
init.m	An initial guess of m for computing MLE of m.
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, 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

`data`	A vector of experimental data (non-negative integers).
`e`	Plating efficiency.
`init.m`	An initial guess of the parameter m.
`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 secent 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.B0 *A Limiting Distribution Induced by the Bartlett Mutation Model*

Description

This discrete distribution, as a limiting distribution induced by the Bartlett model (Bartlett 1955; Zheng 2008), was first discovered by Zheng (2010). This distribution is often denoted by B0(A,k), where A and k are both positive real parameters. It was later found that this distribution can also be regarded as a continuous mixture of the Luria-Delbruck distribution, where the mixing distribution of the parameter m is a gamma distribution (Zheng 2011). Therefore, the relation between the Luria-Delbruck distribution and the B0 distribution is similar to that between the Poisson distribution and the negative binomial distribution.

Usage

```
prob.B0(A, k, n = 5)
```

Arguments

A	A positive real number.
k	A positive real number.
n	An non-negative integer specifying the number of probabilities to be computed.

Value

A vector containing the first n+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).

Q. Zheng, On Bartlett's formulation of the Luria-Delbruck mutation model, Mathematical Biosciences 215 (2008) 48-54.

Q. Zheng, A new discrete distribution induced by the Luria-Delbruck mutation model, Statistics 44 (2010) 529-540.

Q. Zheng, A Bayesian two-level model for fluctuation assay, Genetica 139 (2011) 1409-1416.

See Also

prob.LD, prob.MK

Examples

```
prob.B0(5,0.8,12)
```

`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 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 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)
```

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=(birth rate of mutants)/(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)
```

`samp.size.LD.plating` *Calculating sample size under the Lea-Coulson model with partial plating*

Description

This function calculates required sample size based on rough prior knowledge of m and intended plating efficiency e. The user can estimate the required sample size based on the psi score as defined in Zheng (2017).

Usage

```
samp.size.LD.plating(m = 10.2, e = 0.2, psi = 0.25, trunc = 3000)
```

Arguments

`m`	Rough prior knowledge about m, the expected number of mutations per culture
`e`	The intended plating efficiency, which is strictly between 0 and 1
`psi`	The psi score defined as the half width of a 95% confidence interval for m divided by an anticipated magnitude of m
`trunc`	Number of terms kept in computing expected Fisher information

Value

A positive integer representing required sample size, i.e., the number of cultures

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, Sample size determination for the fluctuation experiment, Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis 795 (2017) 10-14.

See Also

samp.size.MK

Examples

```
samp.size.LD.plating(m=50,e=0.1,psi=0.25)
```

`samp.size.MK` *Calculating sample size under the Mandelbrot-Koch model*

Description

This function calculates required sample size based on rough prior knowledge of m and w. The user can estimate the required sample size based on the psi score as defined in Zheng (2017).

Usage

```
samp.size.MK(m = 1.2, w = 0.9, psi = 0.25, trunc = 3000)
```

Arguments

`m`	Rough prior knowledge about m, the expected number of mutations per culture
`w`	Experimentalist's knowledge about w, relative fitness of mutants
`psi`	The psi score defined as the half width of a 95% confidence interval for m divided by an anticipated magnitude of m
`trunc`	Number of terms kept in computing expected Fisher information

Value

A positive integer representing required sample size, i.e., the number of cultures

Author(s)

Qi Zheng <qzheng@sph.tamhsc.edu>

References

Q. Zheng, Sample size determination for the fluctuation experiment, Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis 795 (2017) 10-14.

See Also

samp.size.LD.plating

Examples

```
samp.size.MK(m=4,w=0.75,psi=0.25)
```

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

Description

This function simulates the dynamics of (Xt,Yt), where Xt is the number of nonmutant cells at time t, and where Yt 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 Biosciences 215:48-54 (2008).

See Also

simu.cultures, simu.Haldane

Examples

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

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)
```

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

Description

There are 13 fluctuation experiments in this data object. From each condensed culture of 2.5ml a sample of 1ml was plated, and hence the plating efficiency is e=0.4. 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, Nt=1.15e9 for the first experiment. See Zheng and Werngren (2018) for further information about the Werngren-Hoffner data.

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.

Q Zheng, J Werngren, An unbiased attitude is vital to exploring the Beijing genotype of Mycobacterium tuberculosis, Tuberculosis 111 (2018) 193-197.

Examples

```
newton.LD.plating(wh.data[[1]],e = 0.4)
```

`wierdl.est` *Wierdl method of accounting for variation in Nt*

Description

Nt is the (non-mutant) cell population size immediately prior to plating. In a fluctuation experiment comprising n tubes, the final cell population sizes are Nt(1), Nt(2), ..., Nt(n). The method proposed by Wierdl et al. (1996) appears to be the first to account for variation in Nt. However, this method requires the experimentalist to measure Nt for all tubes, which is a daunting experimental challenge. The Wierdl method is a generalization of the method of the median devised by Lea and Coulson (1949).

Usage

```
wierdl.est(y, Nt, init.m = 0, tol = 1e-09, max.iter = 25,
   show.iter = FALSE)
```

Arguments

y	A vector containing the mutant counts.
Nt	A vector containing the final cell population sizes.
init.m	An initial guess for m.
tol	Tolerance parameter to control numerical accuracy.
max.iter	A parameter to control the number of iterations in solving the Lea-Coulson equation.
show.iter	To show the iteration process.

Value

A real number as an estimate of the mutation rate, not the usual mean number of mutations (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.

M. Wierdl, C.N. Green, A. Datta, S. Jinks-Robertson, T.D. Petes, Destabilization of simple repetitive DNA sequences by transcription in yeast. Genetics 143 (1996) 713-721.

See Also

newton.B0

Examples

```
Nt=rep(1.9e8, 30); wierdl.est(demerec.data,Nt)
```

Index