Package ‘scModels’

September 3, 2019

Title Fitting Discrete Distribution Models to Count Data

Version 1.0.1

DateNote Previous CRAN version 1.0.0 on 2019-06-13

Maintainer Lisa Amrhein <amrheinlisa@gmail.com>

License GPL-3

Description Provides functions for fitting discrete distribution models to count data.
Included are the Poisson, the negative binomial and, most importantly,
a new implementation of the Poisson-beta distribution (density, distribution and quantile
functions, and random number generator) together with a needed new implementation of
Kummer's function (also: confluent hypergeometric function of the first kind). Three
different implementations of the Gillespie algorithm allow data simulation based on the
basic, switching or bursting mRNA generating processes. Moreover, likelihood functions for
four variants of each of the three aforementioned distributions are also available.
The variants include one population and two population mixtures, both with and without
zero-inflation. The package depends on the 'MPFR' li-
braries (<https://www.mpfr.org/>) which need to be installed separately
(see description at <https://github.com/fuchslab/scModels>).
This package is supplement to the paper ``A mechanistic model for the negative binomial distri-
bution of single-cell mRNA counts''
by Lisa Amrhein, Kumar Harsha and Christiane Fuchs (2019) <doi:10.1101/657619> avail-
able on bioRxiv.

Depends R (>= 3.1.0)

LazyData true

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, testthat

LinkingTo Rcpp

Imports Rcpp

Encoding UTF-8

SystemRequirements gmp (>= 4.2.3), mpfr (>= 3.0.0)

SystemRequirementsNote 'MPFR' (MP Floating-Point Reliable Library,
<http://mpfr.org/>) and 'GMP' (GNU Multiple Precision
library,<http://gmplib.org/>)

NeedsCompilation yes

Author Lisa Amrhein [aut, cre],
Kumar Harsha [aut],
Christiane Fuchs [aut],
Pavel Holoborodko [ctb] (Author and copyright holder of 'mpreal.h')

Repository CRAN

Date/Publication 2019-09-03 00:00:09 UTC

R topics documented:

chf_1F1 *Kummer's (confluent hypergeometric) function*

Description

Kummer's function (also: confluent hypergeometric function of the first kind) for numeric (non-complex) values and input parameters.

Usage

```
chf_1F1(x, a, b)
```

Arguments

`x`	numeric value or vector
`a, b`	numeric parameters of the Kummer function

Examples

```
x <- chf_1F1(-100:100, 5, 7)
plot(-100:100, x, type='l')
```

fit_params — *Functions to estimate parameters of probability distributions by fitting the distributions using optim()*

Description

Functions to estimate parameters of probability distributions by fitting the distributions using optim()

Usage

```
fit_params(x, type, optim_control = list(maxit = 1000))
```

Arguments

`x`	Vector containing the discrete observations
`type`	Keyword for the probability distribution the data is to be fitted against. Possible values are ("pois", "nb", "pb", "pois2", "nb2", "pb2", "zipois", "zinb", "zipb", "zipois2", "zinb2", "zipb2")
`optim_control`	List of options to override presets in the optim function; Set to list(maxit = 1000) by default. For more details, please refer to the 'control' parameter in the standard 'optim' function in package 'stats'.

Examples

```
x1 <- rnbinom(100, size = 13, mu = 9)
p1 <- fit_params(x1, "nb")
s <- sample(x = c(0,1), size = 100, replace = TRUE, prob = c(0.3,0.7))
x2 <- s*x1 + (1-s)*rnbinom(100, size = 15, mu = 53)
p2 <- fit_params(x2, "nb2")
```

gmRNA — *Gillespie algorithm for mRNA generating processes*

Description

Gillespie algorithms allow synthetic data simulation via three different underlying mRNA generating processes: the basic process consists of a simple death-birth model of mRNA transcription and degradation; the switching process considers additionally gene activation and deactivation, with mRNA transcription only happening in active gene states; the bursting process, transcribes mRNA in bursts with geometrically distributed burst sizes.

Usage

```
gmRNA_basic(n, r.on, r.degr)

gmRNA_switch(n, r.act, r.deact, r.on, r.degr)

gmRNA_burst(n, r.burst, s.burst, r.degr)
```

Arguments

`n`	Number of observations
`r.on`	Transcription rate during gene activation (Switching model)
`r.degr`	mRNA degradation rate (all models)
`r.act`	DNA activation rate (Switching Model)
`r.deact`	DNA deactivation rate (Switching Model)
`r.burst`	Bursty transcription rate (Bursting model)
`s.burst`	Mean burst size (Bursting Model)

Examples

```
x <- gmRNA_basic(100, 0.75, 0.001)
plot(density(x))
x <- gmRNA_switch(100, 0.23, 0.15, 0.75, 0.001)
plot(density(x))
x <- gmRNA_burst(10, 0.15, 0.75, 0.001)
plot(density(x))
```

`nlogL` *Negative log Likelihood functions for Poisson, negative binomial and Poisson-beta distributions*

Description

The negative log Likelihood functions for Poisson, negative binomial and Poisson-beta distributions. Mixing two distributions of the same kind and/or adding zero-inflation allows to take characteristics of real data into account. Additionally, one population and two population mixtures - with and without zero-inflations - allow distribution fittings of the Poisson, negative binomial and the Poisson-beta distribution.

Usage

```
nlogL_pois(data, par.pois)

nlogL_nb(data, par.nb)

nlogL_pb(data, par.pb)
```

```
nlogL_pois2(data, par.pois2)

nlogL_nb2(data, par.nb2)

nlogL_pb2(data, par.pb2)

nlogL_zipois(data, par.zipois)

nlogL_zinb(data, par.zinb)

nlogL_zipb(data, par.zipb)

nlogL_zipois2(data, par.zipois2)

nlogL_zinb2(data, par.zinb2)

nlogL_zipb2(data, par.zipb2)
```

Arguments

`data`	Vector containing the discrete observations
`par.pois`	Scalar containing the lambda parameter of the Poisson distribution
`par.nb`	Vector of length 2, containing the size and the mu parameter of the negative binomial distribution
`par.pb`	Vector of length 3, containing the alpha, beta and c parameter of the Poisson-beta distribution
`par.pois2, par.nb2, par.pb2`	Vector containing the parameters of the two mixing distributions. First entry represents the fraction of the first distribution, followed by all parameters of the first, then all of the second distribution.
`par.zipois, par.zinb, par.zipb`	Vector containing the respective zero-inflated distribution parameters. The additional first entry is the inflation parameter for all cases.
`par.zipois2, par.zinb2, par.zipb2`	Parameters for the zero-inflated two population model.

Details

Functions nlogL_pois, nlogL_nb, nlogL_pb compute the negative log-likelihood of Poisson, negative binomial and the Poisson-beta distributions given the data. Functions nlogL_pois2, nlogL_nb2 and nlogL_pb2 compute the negative log-likelihood values for a two population mixture of distributions whereas nlogL_zipois, nlogL_zinb, nlogL_zipb compute the same for the zero-inflated distributions. Furthermore, nlogL_zipois2, nlogL_zinb2 and nlogL_zipb2 are for two population mixtures with zero-inflation.

Examples

```
x <- rpois(100, 11)
```

```
nl1 <- nlogL_pois(x, 11)
nl2 <- nlogL_pois(x, 13)
x <- rnbinom(100, size = 13, mu = 9)
nl <- nlogL_nb(x, c(13, 9))
x <- rpb(n = 1000, alpha=5, beta= 3, c=20)
nl <- nlogL_pb(x, c(5, 3, 20))
s <- sample(x = c(0,1), size = 100, replace = TRUE, prob = c(0.3,0.7))
x <- s*rpois(100, 7) + (1-s)*rpois(100, 13)
nl1 <- nlogL_pois2(x, c(0.7, 13, 7))
nl2 <- nlogL_pois2(x, c(0.3, 7, 13))
## both values should be same: 296.9517
s <- sample(x = c(0,1), size = 100, replace = TRUE, prob = c(0.3,0.7))
x <-s*rnbinom(100, size = 13, mu = 9) + (1-s)*rnbinom(100, size = 17, mu = 29)
nl <- nlogL_nb2(x, c(0.7, 17, 29, 13, 9))
s <- sample(x = c(0,1), size = 100, replace = TRUE, prob = c(0.3,0.7))
x <- s*rpb(100, 5, 3, 20) + (1-s)*rpb(100, 7, 13, 53)
nl <- nlogL_pb2(x, c(0.7, 7, 13, 53, 5, 3, 20))
x <- c(rep(0, 10), rpois(90, 7))
nl <- nlogL_zipois(x, c(0.1, 7))
x <- c(rep(0,10), rnbinom(90, size = 13, mu = 9))
nl <- nlogL_zinb(x, c(0.1, 13, 9))
x <- c(rep(0, 10), rpb(n = 90, alpha=5, beta= 3, c=20))
nl <- nlogL_zipb(x, c(0.1, 5, 3, 20))
s <- sample(x = c(0,1), size = 90, replace = TRUE, prob = c(0.3,0.7))
x <- c(rep(0, 10), s*rpois(90, 7) + (1-s)*rpois(90, 13))
nl1 <- nlogL_zipois2(x, c(0.1, 0.63, 13, 7))
s <- sample(x = c(0,1), size = 90, replace = TRUE, prob = c(0.3,0.7))
x <- c(rep(0, 10), s*rnbinom(90, size = 13, mu = 9) + (1-s)*rnbinom(90, size = 17, mu = 29))
nl <- nlogL_zinb2(x, c(0.1, 0.63, 17, 29, 13, 9))
s <- sample(x = c(0,1), size = 90, replace = TRUE, prob = c(0.3,0.7))
x <- c(rep(0,10), s*rpb(90, 5, 3, 20) + (1-s)*rpb(90, 7, 13, 53))
nl <- nlogL_zipb2(x, c(0.1, 0.63, 7, 13, 53, 5, 3, 20))
```

Poisson-beta *Poisson-beta Distribution*

Description

Density, distribution function, quantile function and random generation for the Poisson-beta distribution: a Poisson distribution whose parameter itself follows a beta distribution. Alpha and beta are the parameters of this specific beta distribution which is scaled on (0, c) in contrast to the usual scaling of the standard beta distribution on (0,1).

Usage

```
dpb(x, alpha, beta, c = 1, log = FALSE)

ppb(q, alpha, beta, c = 1, lower.tail = TRUE, log.p = FALSE)

qpb(p, alpha, beta, c = 1, lower.tail = TRUE, log.p = FALSE)
```

```
rpb(n, alpha, beta, c = 1)
```

Arguments

`x, q`	Vector of (non-negative integer) quantiles
`alpha, beta`	Non-negative parameters of the beta distribution (shape1 and shape2)
`c`	Numeric scaling parameter of the beta distribution. The standard beta is scaled on (0,1) (default) and can be transformed to (0,c).
`log, log.p`	Logical; if TRUE, probabilities p are given as log(p)
`lower.tail`	Logical; if TRUE (default), probabilities are $P[X \le x]$ otherwise, $P[X > x]$.
`p`	Vector of probabilities
`n`	Number of observations

Examples

```
X <- dpb(x=0:200, alpha=5, beta=3, c=20)
plot(0:200, X, type='l')
Y <- dpb(0:10, seq(10.0,11.0,by=0.1), seq(30.0,31.0,by=0.1), seq(10.2,11.2,by=0.1))
Y <- ppb(q= 0 :200, alpha=5, beta= 3, c=20)
plot(0:200, Y, type="l")
Z <- qpb(p= seq(0,1, by= 0.01), alpha=5, beta= 3, c=20)
plot(seq(0,1, by= 0.01),Z, type="l")
RV <- rpb(n = 1000, alpha=5, beta= 3, c=20)
plot(0 : 200, X, type="l")
lines(density(RV), col="red")
R2 <- rpb(11, seq(10.0,11.0,by=0.1), seq(30.0,31.0,by=0.1), seq(10.2,11.2,by=0.1))
```

Index