

Package: scModels (via r-universe)

October 26, 2024

Title Fitting Discrete Distribution Models to Count Data

Version 1.0.4

DateNote Previous CRAN version 1.0.3 on 2022-03-17

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, the Poisson-inverse gaussian 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' libraries (<<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 distribution of single-cell mRNA counts" by Lisa Amrhein, Kumar Harsha and Christiane Fuchs (2019) <[doi:10.1101/657619](https://doi.org/10.1101/657619)> available on bioRxiv.

Depends R (>= 3.1.0)

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, testthat

LinkingTo Rcpp

Imports Rcpp, gamlss.dist

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] (<<https://orcid.org/0000-0002-0370-624X>>), Kumar Harsha [aut] (<<https://orcid.org/0000-0002-3865-5286>>), Christiane Fuchs [aut] (<<https://orcid.org/0000-0003-3565-8315>>), Pavel Holoborodko [ctb] (Author and copyright holder of 'mpreal.h')

Date/Publication 2023-01-24 08:20:02 UTC

Repository <https://lisaamrhein.r-universe.dev>

RemoteUrl <https://github.com/cran/scModels>

RemoteRef HEAD

RemoteSha ba14d424c697de7a21f22f10407afbab9c5aaf7d

Contents

chf_1F1	2
fit_params	3
gmRNA	4
Inverse Gaussian	5
nlogL	5
Poisson-beta	8

Index **10**

chf_1F1	<i>Kummer's (confluent hypergeometric) function in log-scale</i>
---------	--

Description

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

Usage

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

Arguments

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

Details

Note that the output is in log-scale. So the evaluated function is:

$$\log \left[\sum_{n=0}^{\infty} \frac{a^{(n)} x^n}{b^{(n)} n!} \right]$$

where $a^{(n)}$ and $b^{(n)}$ describe the rising factorial.

Examples

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

fit_params	<i>Functions to estimate parameters of probability distributions by fitting the distributions using optim()</i>
------------	---

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", "del", "pig", "pb", "pois2", "nb2", "del2", "pig2", "pb2", "zipois", "zinb", "zidel", "zipg", "zipb", "zipois2", "zinb2", "zidel2", "zipig2", "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 <- rnbino(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) * rnbino(100, size = 15, mu = 53)
p2 <- fit_params(x2, "nb2")
```

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. The basic_burst model combines both the basic and the burst model. The IGbasic burst model describes the basic model with non-constant transcription rates, but transcription rates follow an inverse Gaussian distribution governed by one parameter, the mean parameter of the inverse Gaussian distribution. Additionally a burst transcription occurs (with NB distributed burst sizes), the whole burst (rate and burst sizes) are determined by the rate parameter.

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)
gmRNA_basic_burst(n, r.on, r.burst, s.burst, r.degr)
gmRNA_IGbasic_burst(n, r.mu, r.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, Basic Burst model and IG Basic Burst model)
s.burst	Mean burst size (Bursting Model and Basic Burst model)
r.mu	Mean parameter for the inverse Gaussian distribution (IG Basic Burst 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))
x <- gmRNA_basic_burst(10, 0.75, 0.15, 0.5, 0.001)
plot(density(x))
x <- gmRNA_IGbasic_burst(10, 2, 0.5, 0.1)
plot(density(x))

```

Inverse Gaussian	<i>Inverse Gaussian Distribution</i>
------------------	--------------------------------------

Description

random generation function for the inverse Gaussian distribution: Mu and lambda are the parameters of this distribution.

Usage

```
rInvGaus(n, mu, lambda)
```

Arguments

n	Number of observations
mu, lambda	Non-negative parameters of the inverse Gaussian distribution (mean and shape)

Examples

```
RV <- rInvGaus(n = 100, mu = 10, lambda = 2)
```

nlogL	<i>Negative log Likelihood functions for Poisson, negative binomial, Delaporte, Poisson-inverse Gaussian and Poisson-beta distributions</i>
-------	---

Description

The negative log Likelihood functions for Poisson, negative binomial, Delaporte, Poisson-inverse Gaussian 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-inflation - allow distribution fitting of the Poisson, negative binomial, Delaporte, Poisson-inverse Gaussian and the Poisson-beta distribution.

Usage

```
nlogL_pois(data, par.pois)
nlogL_nb(data, par.nb)
nlogL_del(data, par.del)
nlogL_pig(data, par.pig)
nlogL_pb(data, par.pb)
nlogL_pois2(data, par.pois2)
nlogL_nb2(data, par.nb2)
nlogL_del2(data, par.del2)
nlogL_pig2(data, par.pig2)
nlogL_pb2(data, par.pb2)
nlogL_zipois(data, par.zipois)
nlogL_zinb(data, par.zinb)
nlogL_zidel(data, par.zidel)
nlogL_zipig(data, par.zipig)
nlogL_zipb(data, par.zipb)
nlogL_zipois2(data, par.zipois2)
nlogL_zinb2(data, par.zinb2)
nlogL_zidel2(data, par.zidel2)
nlogL_zipig2(data, par.zipig2)
nlogL_zipb2(data, par.zipb2)
```

Arguments

<code>data</code>	Vector containing the discrete observations
<code>par.pois</code>	Scalar containing the lambda parameter of the Poisson distribution
<code>par.nb</code>	Vector of length 2, containing the size and the mu parameter of the negative binomial distribution

par.del	Vector of length 3, containing the mu, sigma and the nu parameter of the Delaporte distribution
par.pig	Vector of length 2, containing the mu and the sigma parameter of the Poisson-inverse Gaussian distribution
par.pb	Vector of length 3, containing the alpha, beta and c parameter of the Poisson-beta distribution
par.pois2, par.nb2, par.del2, par.pig2, 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.zidel, par.zipig, 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.zidel2, par.zipig2, par.zipb2	Parameters for the zero-inflated two population model.

Details

Functions `nlogL_pois`, `nlogL_nb`, `nlogL_del`, `nlogL_pig`, `nlogL_pb` compute the negative log-likelihood of Poisson, negative binomial, Poisson-inverse Gaussian and the Poisson-beta distributions given the data. Functions `nlogL_pois2`, `nlogL_nb2`, `nlogL_del2`, `nlogL_pig2` and `nlogL_pb2` compute the negative log-likelihood values for a two population mixture of distributions whereas `nlogL_zipois`, `nlogL_zinb`, `nlogL_zidel`, `nlogL_zipig`, `nlogL_zipb` compute the same for the zero-inflated distributions. Furthermore, `nlogL_zipois2`, `nlogL_zinb2`, `nlogL_zidel2`, `nlogL_zipig2` and `nlogL_zipb2` are for two population mixtures with zero-inflation.

Examples

```
x <- rpois(100, 11)
n1 <- nlogL_pois(x, 11)
n2 <- nlogL_pois(x, 13)
x <- rnbinom(100, size = 13, mu = 9)
n1 <- nlogL_nb(x, c(13, 9))
x <- gamlss.dist::rDEL(100, mu = 5, sigma = 0.2, nu = 0.5)
n1 <- nlogL_del(x, c(5, 0.2, 0.5))
x <- gamlss.dist::rPIG(100, mu = 5, sigma = 0.2)
n1 <- nlogL_pig(x, c(5, 0.2))
x <- rpb(n = 1000, alpha = 5, beta = 3, c = 20)
n1 <- 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)
n1 <- nlogL_pois2(x, c(0.3, 13, 7))
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)
n1 <- nlogL_nb2(x, c(0.3, 17, 29, 13, 9))
s <- sample(x = c(0,1), size = 100, replace = TRUE, prob = c(0.3,0.7))
x <- s * gamlss.dist::rDEL(100, mu = 5, sigma = 0.2, nu = 0.5) +
  (1 - s) * gamlss.dist::rDEL(100, mu = 20, sigma = 2, nu = 0.1)
n1 <- nlogL_del2(x, c(0.7,5, 0.2, 20, 2))
s <- sample(x = c(0,1), size = 100, replace = TRUE, prob = c(0.3,0.7))
```

```

x <- s * gamlss.dist::rPIG(100, mu = 5, sigma = 0.2) +
  (1 - s) * gamlss.dist::rPIG(100, mu = 20, sigma = 2)
n1 <- nlogL_pig2(x, c(0.7, 5, 0.2, 20, 2))
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)
n1 <- nlogL_pb2(x, c(0.7, 7, 13, 53, 5, 3, 20))
x <- c(rep(0, 10), rpois(90, 7))
n1 <- nlogL_zipois(x, c(0.1, 7))
x <- c(rep(0,10), rnbinom(90, size = 13, mu = 9))
n1 <- nlogL_zinb(x, c(0.1, 13, 9))
x <- c(rep(0,10), gamlss.dist::rDEL(90, mu = 13, sigma = 2, nu = 0.5))
n1 <- nlogL_zidel(x, c(0.1, 13, 2, 0.5))
x <- c(rep(0,10), gamlss.dist::rPIG(90, mu = 13, sigma = 2))
n1 <- nlogL_zipig(x, c(0.1, 13, 2))
x <- c(rep(0, 10), rpb(n = 90, alpha=5, beta= 3, c=20))
n1 <- 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))
n1 <- nlogL_zipois2(x, c(0.1, 0.63, 7, 13))
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))
n1 <- nlogL_zinb2(x, c(0.1, 0.63, 13, 9, 17, 29))
s <- sample(x = c(0, 1), size = 90, replace = TRUE, prob = c(0.3, 0.7))
x <- c(rep(0, 10), s * gamlss.dist::rDEL(90, mu = 13, sigma = 9, nu = 0.5) +
  (1 - s) * gamlss.dist::rDEL(90, mu = 17, sigma = 29, nu = 0.1))
n1 <- nlogL_zidel2(x, c(0.1, 0.63, 13, 9, 0.5, 17, 29, 0.1))
s <- sample(x = c(0,1), size = 90, replace = TRUE, prob = c(0.3, 0.7))
x <- c(rep(0, 10), s * gamlss.dist::rPIG(90, mu = 13, sigma = 0.2) +
  (1-s) * gamlss.dist::rPIG(90, mu = 17, sigma = 2))
n1 <- nlogL_zipig2(x, c(0.1, 0.63, 13, 0.2, 17, 2))
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))
n1 <- 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 $P[X \leq x]$ 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

- * **Gaussian**
 - Inverse Gaussian, 5
- * **Inverse**
 - Inverse Gaussian, 5
- * **Poisson-beta**
 - nlogL, 5
 - Poisson-beta, 8
- * **binomial**
 - nlogL, 5
- * **distribution**
 - Inverse Gaussian, 5
 - Poisson-beta, 8
- * **estimation**
 - fit_params, 3
- * **likelihood**
 - nlogL, 5
- * **negative**
 - nlogL, 5
- * **parameter**
 - fit_params, 3

chf_1F1, 2

dpb (Poisson-beta), 8

fit_params, 3

gmRNA, 4

- gmRNA_basic (gmRNA), 4
- gmRNA_basic_burst (gmRNA), 4
- gmRNA_burst (gmRNA), 4
- gmRNA_IGbasic_burst (gmRNA), 4
- gmRNA_switch (gmRNA), 4

Inverse Gaussian, 5

nlogL, 5

- nlogL_del (nlogL), 5
- nlogL_del2 (nlogL), 5
- nlogL_nb (nlogL), 5
- nlogL_nb2 (nlogL), 5

- nlogL_pb (nlogL), 5
- nlogL_pb2 (nlogL), 5
- nlogL_pig (nlogL), 5
- nlogL_pig2 (nlogL), 5
- nlogL_pois (nlogL), 5
- nlogL_pois2 (nlogL), 5
- nlogL_zidel (nlogL), 5
- nlogL_zidel2 (nlogL), 5
- nlogL_zinb (nlogL), 5
- nlogL_zinb2 (nlogL), 5
- nlogL_zipb (nlogL), 5
- nlogL_zipb2 (nlogL), 5
- nlogL_zipig (nlogL), 5
- nlogL_zipig2 (nlogL), 5
- nlogL_zipois (nlogL), 5
- nlogL_zipois2 (nlogL), 5

Poisson-beta, 8

ppb (Poisson-beta), 8

qpb (Poisson-beta), 8

rInvGaus (Inverse Gaussian), 5

rpb (Poisson-beta), 8