Package ‘scModels’

February 26, 2021

**Title**  Fitting Discrete Distribution Models to Count Data

**Version**  1.0.2

**DateNote**  Previous CRAN version 1.0.1 on 2019-09-03

**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' 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, 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/>)

1
**chf_1F1**

**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**

```r
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**

```r
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**

```r
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")
```
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

\begin{verbatim}
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)
\end{verbatim}

Arguments

- \textit{n} \hspace{1cm} Number of observations
- \textit{r.on} \hspace{1cm} Transcription rate during gene activation (Switching model)
- \textit{r.degr} \hspace{1cm} mRNA degradation rate (all models)
- \textit{r.act} \hspace{1cm} DNA activation rate (Switching Model)
- \textit{r.deact} \hspace{1cm} DNA deactivation rate (Switching Model)
- \textit{r.burst} \hspace{1cm} Bursty transcription rate (Bursting model, Basic Burst model and IG Basic Burst model)
- \textit{s.burst} \hspace{1cm} Mean burst size (Bursting Model and Basic Burst model)
- \textit{r.mu} \hspace{1cm} Mean parameter for the inverse Gaussian distribution (IG Basic Burst model)

Examples

\begin{verbatim}
x <- gmRNA_basic(100, 0.75, 0.001)
plot(density(x))
x <- gmRNA_switch(100, 0.23, 0.15, 0.75, 0.001)
\end{verbatim}
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**

**Inverse Gaussian Distribution**

**Description**

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

**Usage**

\[
\text{rInvGaus}(n, \mu, \lambda)
\]

**Arguments**

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

**Examples**

\[
\text{RV} \leftarrow \text{rInvGaus}(n = 100, \mu = 10, \lambda = 2)
\]

---

**nlogL**

Negative log Likelihood functions for Poisson, negative binomial, Delaporte, Poisson-inverse Gaussian and Poisson-beta distributions

**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

\texttt{nlogL\_pois(data, par.pois)}
\texttt{nlogL\_nb(data, par.nb)}
\texttt{nlogL\_del(data, par.del)}
\texttt{nlogL\_pig(data, par.pig)}
\texttt{nlogL\_pb(data, par.pb)}
\texttt{nlogL\_pois2(data, par.pois2)}
\texttt{nlogL\_nb2(data, par.nb2)}
\texttt{nlogL\_del2(data, par.del2)}
\texttt{nlogL\_pig2(data, par.pig2)}
\texttt{nlogL\_pb2(data, par.pb2)}
\texttt{nlogL\_zipois(data, par.zipois)}
\texttt{nlogL\_zinb(data, par.zinb)}
\texttt{nlogL\_zidel(data, par.zidel)}
\texttt{nlogL\_zipig(data, par.zipig)}
\texttt{nlogL\_zipb(data, par.zipb)}
\texttt{nlogL\_zipois2(data, par.zipois2)}
\texttt{nlogL\_zinb2(data, par.zinb2)}
\texttt{nlogL\_zidel2(data, par.zidel2)}
\texttt{nlogL\_zipig2(data, par.zipig2)}
\texttt{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
Vector of length 3, containing the mu, sigma and the nu parameter of the Dela-
porte distribution

Vector of length 2, containing the mu and the sigma parameter of the Poisson-
inverse Gaussian distribution

Vector of length 3, containing the alpha, beta and c parameter of the Poisson-
beta distribution

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.

Vector containing the respective zero-inflated distribution parameters. The ad-
ditional first entry is the inflation parameter for all cases.

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

```r
x <- rpois(100, 11)
nl1 <- nlogL_pois(x, 11)
nl2 <- nlogL_pois(x, 13)
x <- rnbinom(100, size = 13, mu = 9)
ln <- nlogL_nb(x, c(13, 9))
x <- gamlss.dist::rDEL(100, mu = 5, sigma = 0.2, nu = 0.5)
ln <- nlogL_del(x, c(5, 0.2, 0.5))
x <- gamlss.dist::rPIG(100, mu = 5, sigma = 0.2)
ln <- nlogL_pig(x, c(5, 0.2))
x <- rpb(n = 1000, alpha = 5, beta = 3, c = 20)
ln <- 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)
ln <- 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)
ln <- 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)
ln <- 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))
```
Poisson-beta

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
rInvGaus (Inverse Gaussian), 5