Package ‘zipfextR’

September 18, 2019

Type Package

Title Zipf Extended Distributions

Version 1.0.1

Author Ariel Duarte-López [aut, cre] (<https://orcid.org/0000-0002-7432-0344>), Marta Pérez-Casany [aut] (<https://orcid.org/0000-0003-3675-6902>)

Maintainer Ariel Duarte-López <aduarte@ac.upc.edu>

Description Implementation of four extensions of the Zipf distribution: the Marshall-Olkin Extended Zipf (MOEZipf) Pérez-Casany, M., & Casellas, A. (2013) <arXiv:1304.4540>, the Zipf-Poisson Extreme (Zipf-PE), the Zipf-Poisson Stopped Sum (Zipf-PSS) and the Zipf-Polylog distributions. In log-log scale, the two first extensions allow for top-concavity and top-convexity while the third one only allows for top-concavity. All the extensions maintain the linearity associated with the Zipf model in the tail.

License GPL-3

Depends R (>= 2.0.1)

Imports VGAM (>= 0.9.8), tolerance(>= 1.2.0), copula(>= 0.999-18)

Encoding UTF-8

LazyData true

URL https://github.com/ardlop/zipfextR

BugReports https://github.com/ardlop/zipfextR/issues

RoxygenNote 6.1.1

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2019-09-18 14:40:02 UTC
getInitialValues

getInitialValues                   2
moezipf                           4
moezipfFit                        5
moezipfMean                      7
moezipfMoments                   8
moezipfVariance                  9
zipfpe                           9
zipfpeFit                        11
zipfpeMean                      13
zipfpeMoments                   14
zipfpeVariance                 14
zipfPolylog                     15
zipfPolylogFit                 16
zipfPss                          17
zipfPssFit                      18
zipfPssMean                    20
zipfPssMoments                 21
zipfPssVariance               22
zi_zipfPss                    23
zi_zipfPssFit                24

Index

getInitialValues               26

getInitialValues

Calculates initial values for the parameters of the models.

Description

The selection of appropriate initial values to compute the maximum likelihood estimations reduces the number of iterations which in turn, reduces the computation time. The initial values proposed by this function are computed using the first two empirical frequencies.

Usage

getInitialValues(data, model = "zipf")

Arguments

data    Matrix of count data.
model   Specify the model that requests the initial values (default='zipf').
getInitialValues

Details

The argument data is a two column matrix with the first column containing the observations and the second column containing their frequencies. The argument model refers to the selected model of those implemented in the package. The possible values are: zipf, moezipf, zipfpe, zipfpss or its zero truncated version zt_zipfpss. By default, the selected model is the Zipf one.

For the MOEZipf, the Zipf-PE and the zero truncated Zipf-PSS models that contain the Zipf model as a particular case, the \( \beta \) value will correspond to the one of the Zipf model (i.e. \( \beta = 1 \) for the MOEZipf, \( \beta = 0 \) for the Zipf-PE and \( \lambda = 0 \) for the zero truncated Zipf-PSS model) and the initial value for \( \alpha \) is set to be equal to:

\[
\alpha_0 = \log_2 \left( \frac{f_r(1)}{f_r(2)} \right),
\]

where \( f_r(1) \) and \( f_r(2) \) are the empirical relative frequencies of one and two. This value is obtained equating the two empirical probabilities to their theoretical ones.

For the case of the Zipf-PSS the proposed initial values are obtained equating the empirical probability of zero to the theoretical one which gives:

\[
\lambda_0 = -\log(f_r(0)),
\]

where \( f_r(0) \) is the empirical relative frequency of zero. The initial value of \( \alpha \) is obtained equating the ratio of the theoretical probabilities at zero and one to the empirical ones. This gives place to:

\[
\alpha_0 = \zeta^{-1}(\lambda_0 + f_r(0)/f_r(1)),
\]

where \( f_r(0) \) and \( f_r(1) \) are the empirical relative frequencies associated to the values 0 and 1 respectively. The inverse of the Riemann Zeta function is obtained using the optim routine.

Value

Returns the initial values of the parameters for a given distribution.

References


Examples

data <- rmoezipf(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(levels(data[,1])[data[,1]])
initials <- getInitialValues(data, model='zipf')
Description

Probability mass function, cumulative distribution function, quantile function and random number generation for the MOEZipf distribution with parameters $\alpha$ and $\beta$. The support of the MOEZipf distribution are the strictly positive integer numbers large or equal than one.

Usage

dmoezipf(x, alpha, beta, log = FALSE)
pmoezipf(q, alpha, beta, log.p = FALSE, lower.tail = TRUE)
qmoezipf(p, alpha, beta, log.p = FALSE, lower.tail = TRUE)
rmoezipf(n, alpha, beta)

Arguments

- $x, q$ Vector of positive integer values.
- alpha Value of the $\alpha$ parameter ($\alpha > 1$).
- beta Value of the $\beta$ parameter ($\beta > 0$).
- 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 random values to return.

Details

The probability mass function at a positive integer value $x$ of the MOEZipf distribution with parameters $\alpha$ and $\beta$ is computed as follows:

$$p(x|\alpha, \beta) = \frac{x^{-\alpha} \beta \zeta(\alpha)}{[\zeta(\alpha) - \beta \zeta(\alpha, x)][\zeta(\alpha) - \beta \zeta(\alpha, x + 1)]}, \quad x = 1, 2, \ldots, \alpha > 1, \beta > 0,$$

where $\zeta(\alpha)$ is the Riemann-zeta function at $\alpha$, $\zeta(\alpha, x)$ is the Hurwitz zeta function with arguments $\alpha$ and $x$, and $\bar{\beta} = 1 - \beta$.

The cumulative distribution function, at a given positive integer value $x$, is computed as $F(x) = 1 - S(x)$, where the survival function $S(x)$ is equal to:

$$S(x) = \frac{\beta \zeta(\alpha, x + 1)}{\zeta(\alpha) - \beta \zeta(\alpha, x + 1)}, \quad x = 1, 2, \ldots.$$
The quantile of the MOEZipf($\alpha, \beta$) distribution of a given probability value $p$ is equal to the quantile of the Zipf($\alpha$) distribution at the value:

$$p' = \frac{p \beta}{1 + p (\beta - 1)}$$

The quantiles of the Zipf($\alpha$) distribution are computed by means of the `tolerance` package.

To generate random data from a MOEZipf one applies the `quantile` function over $n$ values randomly generated from an Uniform distribution in the interval (0, 1).

**Value**

dmoezipf gives the probability mass function, pmoezipf gives the cumulative distribution function, qmoezipf gives the quantile function, and rmoezipf generates random values from a MOEZipf distribution.

**References**


**Examples**

dmoezipf(1:10, 2.5, 1.3)
pmoezipf(1:10, 2.5, 1.3)
qmoezipf(0.56, 2.5, 1.3)
rmoezipf(10, 2.5, 1.3)

**Description**

For a given sample of strictly positive integer numbers, usually of the type of ranking data or frequencies of frequencies data, estimates the parameters of the MOEZipf distribution by means of the maximum likelihood method. The input data should be provided as a frequency matrix.
Usage

moezipfFit(data, init_alpha = NULL, init_beta = NULL, level = 0.95, 
...)

## S3 method for class 'moezipfR'
residuals(object, ...)

## S3 method for class 'moezipfR'
fitted(object, ...)

## S3 method for class 'moezipfR'
coef(object, ...)

## S3 method for class 'moezipfR'
plot(x, ...)

## S3 method for class 'moezipfR'
print(x, ...)

## S3 method for class 'moezipfR'
summary(object, ...)

## S3 method for class 'moezipfR'
logLik(object, ...)

## S3 method for class 'moezipfR'
AIC(object, ...)

## S3 method for class 'moezipfR'
BIC(object, ...)

Arguments

data     Matrix of count data in form of a table of frequencies.
init_alpha Initial value of \( \alpha \) parameter \((\alpha > 1)\).
init_beta Initial value of \( \beta \) parameter \((\beta > 0)\).
level     Confidence level used to calculate the confidence intervals (default 0.95).
...        Further arguments to the generic functions. The extra arguments are passing to 
the optim function.
object    An object from class "moezipfR" (output of moezipfFit function).
x         An object from class "moezipfR" (output of moezipfFit function).

Details

The argument data is a two column matrix with the first column containing the observations and 
the second column containing their frequencies.
The log-likelihood function is equal to:

\[
l(\alpha, \beta; x) = -\alpha \sum_{i=1}^{m} f_a(x_i) \log(x_i) + N(\log(\beta) + \log(\zeta(\alpha))) \\
- \sum_{i=1}^{m} f_a(x_i) \log[(\zeta(\alpha) - \bar{\beta}\zeta(\alpha, x_i)(\zeta(\alpha) - \bar{\beta}\zeta(\alpha, x_i + 1))],
\]

where \( f_a(x_i) \) is the absolute frequency of \( x_i \), \( m \) is the number of different values in the sample and \( N \) is the sample size, i.e. \( N = \sum_{i=1}^{m} x_i f_a(x_i) \).

By default the initial values of the parameters are computed using the function `getInitialValues`. The function `optim` is used to estimate the parameters.

**Value**

Returns a `moezipfR` object composed by the maximum likelihood parameter estimations jointly with their standard deviation and confidence intervals. It also contains the value of the log-likelihood at the maximum likelihood estimator.

**See Also**

`getInitialValues`.

**Examples**

```r
data <- rmoezipf(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(as.character(data[,1]))
data[,2] <- as.numeric(as.character(data[,2]))
initValues <- getInitialValues(data, model='moezipf')
obj <- moezipfFit(data, init_alpha = initValues$init_alpha, init_beta = initValues$init_beta)
```

### `moezipfMean`

**Expected value.**

**Description**

Computes the expected value of the MOEZipf distribution for given values of parameters \( \alpha \) and \( \beta \).

**Usage**

```r
moezipfMean(alpha, beta, tolerance = 10^(-4))
```

**Arguments**

- **alpha**: Value of the \( \alpha \) parameter (\( \alpha > 2 \)).
- **beta**: Value of the \( \beta \) parameter (\( \beta > 0 \)).
- **tolerance**: Tolerance used in the calculations (default = \( 10^{-4} \)).
Details

The mean of the distribution only exists for $\alpha$ strictly greater than 2. It is computed by calculating the partial sums of the serie, and stopping when two consecutive partial sums differ less than the tolerance value. The value of the last partial sum is returned.

Value

A positive real value corresponding to the mean value of the distribution.

Examples

\[
\text{moezipfMean}(2.5, 1.3) \\
moezipfMean(2.5, 1.3, 10^{-3})
\]

Description

General function to compute the k-th moment of the MOEZipf distribution for any integer value $k \geq 1$, when it exists. The k-th moment exists if and only if $\alpha > k + 1$. For $k = 1$, this function returns the same value as the moezipfMean function.

Usage

\[
\text{moezipfMoments}(k, \alpha, \beta, \text{tolerance} = 10^{-4})
\]

Arguments

- **k**: Order of the moment to compute.
- **alpha**: Value of the $\alpha$ parameter ($\alpha > k + 1$).
- **beta**: Value of the $\beta$ parameter ($\beta > 0$).
- **tolerance**: Tolerance used in the calculations (default = $10^{-4}$).

Details

The k-th moment is computed by calculating the partial sums of the serie, and stopping when two consecutive partial sums differ less than the tolerance value. The value of the last partial sum is returned.

Value

A positive real value corresponding to the k-th moment of the distribution.

Examples

\[
\text{moezipfMoments}(3, 4.5, 1.3) \\
moezipfMoments(3, 4.5, 1.3, 1*10^{-3})
\]
moezipfVariance

Variance of the MOEZipf distribution.

Description
Computes the variance of the MOEZipf distribution for given values of $\alpha$ and $\beta$.

Usage
moezipfVariance(alpha, beta, tolerance = 10^(-4))

Arguments
- alpha: Value of the $\alpha$ parameter ($\alpha > 3$).
- beta: Value of the $\beta$ parameter ($\beta > 0$).
- tolerance: Tolerance used in the calculations. (default = $10^{-4}$)

Details
The variance of the distribution only exists for $\alpha$ strictly greater than 3.

Value
A positive real value corresponding to the variance of the distribution.

See Also
moezipfMoments, moezipfMean.

Examples
moezipfVariance(3.5, 1.3)

zipfpe
The Zipf-Poisson Extreme Distribution (Zipf-PE).

Description
Probability mass function, cumulative distribution function, quantile function and random number generation for the Zipf-PE distribution with parameters $\alpha$ and $\beta$. The support of the Zipf-PE distribution are the strictly positive integer numbers large or equal than one.
Usage

dzipfpe(x, alpha, beta, log = FALSE)
pzipfpe(q, alpha, beta, log.p = FALSE, lower.tail = TRUE)
qzipfpe(p, alpha, beta, log.p = FALSE, lower.tail = TRUE)
rzipfpe(n, alpha, beta)

Arguments

x, q Vector of positive integer values.
alpha Value of the $\alpha$ parameter ($\alpha > 1$).
beta Value of the $\beta$ parameter ($\beta \in (-\infty, +\infty)$).
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 random values to return.

details

The probability mass function of the Zipf-PE distribution with parameters $\alpha$ and $\beta$ at a positive integer value $x$ is computed as follows:

$$p(x|\alpha, \beta) = \frac{e^{\beta(1 - \frac{\zeta(\alpha,x)}{\zeta(\alpha)})}}{e^\beta - 1}, x = 1, 2, \ldots, \alpha > 1, -\infty < \beta < +\infty,$$

where $\zeta(\alpha)$ is the Riemann-zeta function at $\alpha$, and $\zeta(\alpha, x)$ is the Hurtwitz zeta function with arguments $\alpha$ and $x$.

The cumulative distribution function at a given positive integer value $x$, $F(x)$, is equal to:

$$F(x) = \frac{e^{\beta(1 - \frac{\zeta(\alpha,x+1)}{\zeta(\alpha)})}}{e^\beta - 1} - 1$$

The quantile of the Zipf-PE($\alpha, \beta$) distribution of a given probability value $p$ is equal to the quantile of the Zipf($\alpha$) distribution at the value:

$$p^* = \frac{\log(p(e^\beta - 1) + 1)}{\beta}$$

The quantiles of the Zipf($\alpha$) distribution are computed by means of the tolerance package.

To generate random data from a Zipf-PE one applies the quantile function over $n$ values randomly generated from an Uniform distribution in the interval (0, 1).
Value

dzipfpe gives the probability mass function, pzipfpe gives the cumulative function, qzipfpe gives the quantile function, and rzipfpe generates random values from a Zipf-PE distribution.

References


Examples

dzipfpe(1:10, 2.5, -1.5)
pzipfpe(1:10, 2.5, -1.5)
quzipfpe(0.56, 2.5, 1.3)
rzipfpe(10, 2.5, 1.3)

zipfpeR

Zipf-PE parameters estimation.

Description

For a given sample of strictly positive integer values, usually of the type of ranking data or frequencies of frequencies data, estimates the parameters of the Zipf-PE distribution by means of the maximum likelihood method. The input data should be provided as a frequency matrix.

Usage

zipfpeFit(data, init_alpha = NULL, init_beta = NULL, level = 0.95, 
...)

## S3 method for class 'zipfpeR'
residuals(object, ...)

## S3 method for class 'zipfpeR'
fitted(object, ...)

## S3 method for class 'zipfpeR'
coef(object, ...)

## S3 method for class 'zipfpeR'
plot(x, ...)

## S3 method for class 'zipfpeR'
print(x, ...)
summary(object, ...)

## S3 method for class 'zipfpeR'
logLik(object, ...)

## S3 method for class 'zipfpeR'
AIC(object, ...)

## S3 method for class 'zipfpeR'
BIC(object, ...)

Arguments

- **data** Matrix of count data in form of table of frequencies.
- **init_alpha** Initial value of \( \alpha \) parameter (\( \alpha > 1 \)).
- **init_beta** Initial value of \( \beta \) parameter (\( \beta \in (-\infty, +\infty) \)).
- **level** Confidence level used to calculate the confidence intervals (default 0.95).
- **...** Further arguments to the generic functions. The extra arguments are passing to the `optim` function.
- **object** An object from class "zpeR" (output of `zipfpeFit` function).
- **x** An object from class "zpeR" (output of `zipfpeFit` function).

Details

The argument `data` is a two column matrix with the first column containing the observations and the second column containing their frequencies.

The log-likelihood function is equal to:

\[
l(\alpha, \beta; x) = \beta (N - \zeta(\alpha)^{-1} \sum_{i=1}^{m} f_a(x_i) \zeta(\alpha, x_i)) + \sum_{i=1}^{m} f_a(x_i) \log \left( \frac{e^{\beta x_i^{-\alpha}}}{e^\beta - 1} \right),
\]

where \( f_a(x_i) \) is the absolute frequency of \( x_i \), \( m \) is the number of different values in the sample and \( N \) is the sample size, i.e. \( N = \sum_{i=1}^{m} x_i f_a(x_i) \).

By default the initial values of the parameters are computed using the function `getInitialValues`.

The function `optim` is used to estimate the parameters.

Value

Returns an object composed by the maximum likelihood parameter estimations jointly with their standard deviation and confidence intervals. It also contains the value of the log-likelihood at the maximum likelihood estimator.

See Also

`getInitialValues`.
Examples

```r
data <- rzipfpe(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(as.character(data[,1]))
data[,2] <- as.numeric(as.character(data[,2]))
initValues <- getInitialValues(data, model='zipfpe')
obj <- zipfpeFit(data, init_alpha = initValues$init_alpha, init_beta = initValues$init_beta)
```

zipfpeMean

*Expected value of the Zipf-PE distribution.*

Description

Computes the expected value of the Zipf-PE distribution for given values of parameters $\alpha$ and $\beta$.

Usage

```r
zipfpeMean(alpha, beta, tolerance = 10^(-4))
```

Arguments

- `alpha`: Value of the $\alpha$ parameter ($\alpha > 2$).
- `beta`: Value of the $\beta$ parameter ($\beta \in (-\infty, +\infty)$).
- `tolerance`: Tolerance used in the calculations (default = $10^{-4}$).

Details

The mean of the distribution only exists for $\alpha$ strictly greater than 2. It is computed by calculating the partial sums of the serie, and stopping when two consecutive partial sums differ less than the tolerance value. The value of the last partial sum is returned.

Value

A positive real value corresponding to the mean value of the Zipf-PE distribution.

Examples

```r
zipfpeMean(2.5, 1.3)
zipfpeMean(2.5, 1.3, 10^(-3))
```
Description

General function to compute the k-th moment of the Zipf-PE distribution for any integer value \( k \geq 1 \), when it exists. The k-th moment exists if and only if \( \alpha > k + 1 \). For \( k = 1 \), this function returns the same value as the \( \text{zipfpeMean} \) function.

Usage

\[
\text{zipfpeMoments}(k, \alpha, \beta, \text{tolerance} = 10^{-4})
\]

Arguments

- \( k \) Order of the moment to compute.
- \( \alpha \) Value of the \( \alpha \) parameter (\( \alpha > k + 1 \)).
- \( \beta \) Value of the \( \beta \) parameter (\( \beta \in (-\infty, +\infty) \)).
- \( \text{tolerance} \) Tolerance used in the calculations (default = \( 10^{-4} \)).

Details

The k-th moment of the Zipf-PE distribution is finite for \( \alpha \) values strictly greater than \( k + 1 \). It is computed by calculating the partial sums of the series and stopping when two consecutive partial sums differ less than the \( \text{tolerance} \) value. The value of the last partial sum is returned.

Value

A positive real value corresponding to the k-th moment of the distribution.

Examples

\[
\begin{align*}
\text{zipfpeMoments}(3, 4.5, 1.3) \\
\text{zipfpeMoments}(3, 4.5, 1.3, 1*10\text{^(-3)})
\end{align*}
\]

zipfpeVariance

Variance of the Zipf-PE distribution.

Description

Computes the variance of the Zipf-PE distribution for given values of \( \alpha \) and \( \beta \).

Usage

\[
\text{zipfpeVariance}(\alpha, \beta, \text{tolerance} = 10\text{^(-4)})
\]
Arguments
alpha Value of the \( \alpha \) parameter \( (\alpha > 3) \).
beta Value of the \( \beta \) parameter \( (\beta \in (\infty, +\infty)) \).
tolerance Tolerance used in the calculations. (default = \( 10^{-4} \))

Details
The variance of the distribution only exists for \( \alpha \) strictly greater than 3.

Value
A positive real value corresponding to the variance of the distribution.

See Also
zipfpeMoments, zipfpeMean.

Examples
zipfpeVariance(3.5, 1.3)

---

zipfPolylog

The Zipf-Polylog Distribution (Zipf-Polylog).

Description
Probability mass function of the Zipf-Polylog distribution with parameters \( \alpha \) and \( \beta \). The support of the Zipf-Polylog distribution are the strictly positive integer numbers large or equal than one.

Usage
dzipfpolylog(x, alpha, beta, log = FALSE, nSum = 1000)

Arguments
x Vector of positive integer values.
alpha Value of the \( \alpha \) parameter \( (\alpha > 1) \).
beta Value of the \( \beta \) parameter \( (\beta > 0) \).
log Logical; if TRUE, probabilities p are given as log(p).
nSum The number of terms used for computing the Poliologarithm function [Default = 1000].

Details
The \textit{probability mass function} at a positive integer value \( x \) of the Zipf-Polylog distribution with parameters \( \alpha \) and \( \beta \) is computed as follows:
zipfPolylogFit

Value

dzipfpolylog gives the probability mass function

Examples

dzipfpolylog(1:10, 1.61, 0.98)

zipfPolylogFit  ZipfPolylog parameters estimation.

Description

For a given sample of strictly positive integer numbers, usually of the type of ranking data or frequencies of frequencies data, estimates the parameters of the ZipfPolylog distribution by means of the maximum likelihood method. The input data should be provided as a frequency matrix.

Usage

zipfPolylogFit(data, init_alpha, init_beta, level = 0.95, ...)

## S3 method for class 'zipfPolyR'
residuals(object, ...)

## S3 method for class 'zipfPolyR'
fitted(object, ...)

## S3 method for class 'zipfPolyR'
coef(object, ...)

## S3 method for class 'zipfPolyR'
plot(x, ...)

## S3 method for class 'zipfPolyR'
print(x, ...)

## S3 method for class 'zipfPolyR'
summary(object, ...)

## S3 method for class 'zipfPolyR'
logLik(object, ...)

## S3 method for class 'zipfPolyR'
AIC(object, ...)

## S3 method for class 'zipfPolyR'
BIC(object, ...)
Arguments

data Matrix of count data in form of a table of frequencies.
init_alpha Initial value of $\alpha$ parameter ($\alpha > 1$).
init_beta Initial value of $\beta$ parameter ($\beta > 0$).
level Confidence level used to calculate the confidence intervals (default 0.95).
... Further arguments to the generic functions. The extra arguments are passing to the optim function.
object An object from class "zipfPolyR" (output of zipfPolylogFit function).
x An object from class "zipfPolyR" (output of zipfPolylogFit function).

details
The argument data is a two column matrix with the first column containing the observations and the second column containing their frequencies.
The log-likelihood function is equal to:
The function optim is used to estimate the parameters.

Value
Returns a zipfPolyR object composed by the maximum likelihood parameter estimations jointly with their standard deviation and confidence intervals. It also contains the value of the log-likelihood at the maximum likelihood estimator.

zipfpss The Zipf-Poisson Stop Sum Distribution (Zipf-PSS).

Description
Probability mass function, cumulative distribution function, quantile function and random number generation for the Zipf-PSS distribution with parameters $\alpha$ and $\lambda$. The support of the Zipf-PSS distribution are the positive integer numbers including the zero value. In order to work with its zero-truncated version the parameter isTruncated should be equal to True.

Usage

dzipfpss(x, alpha, lambda, log = FALSE, isTruncated = FALSE)
pzipfpss(q, alpha, lambda, log.p = FALSE, lower.tail = TRUE, isTruncated = FALSE)
rzipfpss(n, alpha, lambda, log.p = FALSE, lower.tail = TRUE, isTruncated = FALSE)
qzipfpss(p, alpha, lambda, log.p = FALSE, lower.tail = TRUE, isTruncated = FALSE)
Arguments

- **x, q**: Vector of positive integer values.
- **alpha**: Value of the $\alpha$ parameter ($\alpha > 1$).
- **lambda**: Value of the $\lambda$ parameter ($\lambda > 0$).
- **log, log.p**: Logical; if TRUE, probabilities p are given as log(p).
- **isTruncated**: Logical; if TRUE, the zero truncated version of the distribution is returned.
- **lower.tail**: Logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
- **n**: Number of random values to return.
- **p**: Vector of probabilities.

Details

The support of the $\lambda$ parameter increases when the distribution is truncated at zero being $\lambda \geq 0$. It has been proved that when $\lambda = 0$ one has the degenerated version of the distribution at one.

References


---

**zipfpssFit**

*Zipf-PSS parameters estimation.*

Description

For a given sample of strictly positive integer numbers, usually of the type of ranking data or frequencies of frequencies data, estimates the parameters of the Zipf-PSS distribution by means of the maximum likelihood method. The input data should be provided as a frequency matrix.

Usage

```r
zipfpssFit(data, init_alpha = NULL, init_lambda = NULL, level = 0.95,
            isTruncated = FALSE, ...)
```

```r
## S3 method for class 'zipfpssR'
residuals(object, isTruncated = FALSE, ...)
```

```r
## S3 method for class 'zipfpssR'
fitted(object, isTruncated = FALSE, ...)
```

```r
## S3 method for class 'zipfpssR'
coef(object, ...)
```
## S3 method for class 'zipfpssR'
plot(x, isTruncated = FALSE, ...)

## S3 method for class 'zipfpssR'
print(x, ...)

## S3 method for class 'zipfpssR'
summary(object, isTruncated = FALSE, ...)

## S3 method for class 'zipfpssR'
logLik(object, ...)

## S3 method for class 'zipfpssR'
AIC(object, ...)

## S3 method for class 'zipfpssR'
BIC(object, ...)

Arguments

- **data**: Matrix of count data in form of table of frequencies.
- **init_alpha**: Initial value of \( \alpha \) parameter (\( \alpha > 1 \)).
- **init_lambda**: Initial value of \( \lambda \) parameter (\( \lambda > 0 \)).
- **level**: Confidence level used to calculate the confidence intervals (default 0.95).
- **isTruncated**: Logical; if TRUE, the truncated version of the distribution is returned (default = FALSE)
- **...**: Further arguments to the generic functions. The extra arguments are passing to the `optim` function.
- **object**: An object from class "zpssR" (output of `zipfpssFit` function).
- **x**: An object from class "zpssR" (output of `zipfpssFit` function).

Details

The argument `data` is a two column matrix with the first column containing the observations and the second column containing their frequencies.

The log-likelihood function is equal to:

\[
l(\alpha, \lambda, x) = \sum_{i=1}^{m} f_{a}(x_i) \log(P(Y = x_i)),
\]

where \( m \) is the number of different values in the sample, being \( f_{a}(x_i) \) is the absolute frequency of \( x_i \). The probabilities are calculated applying the Panjer recursion. By default the initial values of the parameters are computed using the function `getInitialValues`. The function `optim` is used to estimate the parameters.
zipfPSSMean

Value

Returns a zipfPSS object composed by the maximum likelihood parameter estimations jointly with their standard deviation and confidence intervals and the value of the log-likelihood at the maximum likelihood estimator.

References


See Also

getInitialValues.

Examples

data <- rzipfPSS(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(as.character(data[,1]))
data[,2] <- as.numeric(as.character(data[,2]))
initValues <- getInitialValues(data, model='zipfPSS')
obj <- zipfPSSFit(data, init_alpha = initValues$init_alpha, init_lambda = initValues$init_lambda)

zipfPSSMean

Expected value of the Zipf-PSS distribution.

Description

Computes the expected value of the Zipf-PSS distribution for given values of parameters $\alpha$ and $\lambda$.

Usage

zipfPSSMean(alpha, lambda, isTruncated = FALSE)

Arguments

alpha Value of the $\alpha$ parameter ($\alpha > 2$).

lambda Value of the $\lambda$ parameter ($\lambda > 0$).

isTruncated Logical; if TRUE Use the zero-truncated version of the distribution to calculate the expected value (default = FALSE).
Details

The expected value of the Zipf-PSS distribution only exists for $\alpha$ values strictly greater than 2. The value is obtained from the law of total expectation that says that:

$$E[Y] = E[N]E[X],$$

where $E[X]$ is the mean value of the Zipf distribution and $E[N]$ is the expected value of a Poisson one. From where one has that:

$$E[Y] = \lambda \frac{\zeta(\alpha - 1)}{\zeta(\alpha)}.$$

Particularly, if one is working with the zero-truncated version of the Zipf-PSS distribution. This values is computed as:

$$E[Y^{ZT}] = \frac{\lambda \zeta(\alpha - 1)}{\zeta(\alpha) (1 - e^{-\lambda})}.$$

Value

A positive real value corresponding to the mean value of the distribution.

References


Examples

```
zipfpssMean(2.5, 1.3)
zipfpssMean(2.5, 1.3, TRUE)
```

---

Description

General function to compute the k-th moment of the Zipf-PSS distribution for any integer value $k \geq 1$, when it exists. The k-th moment exists if and only if $\alpha > k + 1$.

Usage

```
zipfpssMoments(k, alpha, lambda, isTruncated = FALSE, 
                tolerance = 10^(-4))
```

Arguments

- **k**: Order of the moment to compute.
- **alpha**: Value of the $\alpha$ parameter ($\alpha > k + 1$).
- **lambda**: Value of the $\lambda$ parameter ($\lambda > 0$).
- **isTruncated**: Logical; if TRUE, the truncated version of the distribution is returned.
- **tolerance**: Tolerance used in the calculations (default = $10^{-4}$).
Details

The k-th moment of the Zipf-PSS distribution is finite for \( \alpha \) values strictly greater than \( k + 1 \). It is computed by calculating the partial sums of the serie, and stopping when two consecutive partial sums differ less than the tolerance value. The value of the last partial sum is returned.

Value

A positive real value corresponding to the k-th moment of the distribution.

Examples

```r
zipfpssMoments(1, 2.5, 2.3)
zipfpssMoments(1, 2.5, 2.3, TRUE)
```

---

zipfpssVariance

Variance of the Zipf-PSS distribution.

Description

Computes the variance of the Zipf-PSS distribution for given values of parameters \( \alpha \) and \( \lambda \).

Usage

```r
zipfpssVariance(alpha, lambda, isTruncated = FALSE)
```

Arguments

- `alpha`: Value of the \( \alpha \) parameter (\( \alpha > 3 \)).
- `lambda`: Value of the \( \lambda \) parameter (\( \lambda > 0 \)).
- `isTruncated`: Logical; if TRUE Use the zero-truncated version of the distribution to calculate the expected value (default = FALSE).

Details

The variance of the Zipf-PSS distribution only exists for \( \alpha \) values strictly greater than 3. The value is obtained from the law of total variance that says that:

\[
\]

where \( X \) follows a Zipf distribution with parameter \( \alpha \), and \( N \) follows a Poisson distribution with parameter \( \lambda \). From where one has that:

\[
Var[Y] = \lambda \frac{\zeta(\alpha - 2)}{\zeta(\alpha)}
\]

Particularly, if one is working with the zero-truncated version of the Zipf-PSS distribution. This values is computed as:

\[
Var[Y^{ZT}] = \frac{\lambda \zeta(\alpha) \zeta(\alpha - 2) (1 - e^{-\lambda}) - \lambda^2 \zeta(\alpha - 1)^2 e^{-\lambda}}{\zeta(\alpha)^2 (1 - e^{-\lambda})^2}
\]
Value
A positive real value corresponding to the variance of the distribution.

References

Examples

```r
dzipfpssVariance(4.5, 2.3)
dzipfpssVariance(4.5, 2.3, TRUE)
```

Description
Probability mass function for the zero inflated Zipf-PSS distribution with parameters $\alpha$, $\lambda$ and $w$. The support of the zero inflated Zipf-PSS distribution are the positive integer numbers including the zero value.

Usage

```r
d_zi_zipfpss(x, alpha, lambda, w, log = FALSE)
```

Arguments

- `x` Vector of positive integer values.
- `alpha` Value of the $\alpha$ parameter ($\alpha > 1$).
- `lambda` Value of the $\lambda$ parameter ($\lambda > 0$).
- `w` Value of the $w$ parameter ($0 < w < 1$).
- `log` Logical; if TRUE, probabilities p are given as log(p).

Details
The support of the $\lambda$ parameter increases when the distribution is truncated at zero being $\lambda \geq 0$. It has been proved that when $\lambda = 0$ one has the degenerated version of the distribution at one.

References

Zero Inflated Zipf-PSS parameters estimation.

Description

For a given sample of strictly positive integer numbers, usually of the type of ranking data or frequencies of frequencies data, estimates the parameters of the zero inflated Zipf-PSS distribution by means of the maximum likelihood method. The input data should be provided as a frequency matrix.

Usage

zi_zipfpssFit(data, init_alpha = 1.5, init_lambda = 1.5, init_w = 0.1, level = 0.95, ...)

Arguments

data Matrix of count data in form of table of frequencies.
init_alpha Initial value of $\alpha$ parameter ($\alpha > 1$).
init_lambda  Initial value of $\lambda$ parameter ($\lambda > 0$).
init_w    Initial value of $w$ parameter ($0 < w < 1$).
level    Confidence level used to calculate the confidence intervals (default 0.95).
...     Further arguments to the generic functions. The extra arguments are passing to the `optim` function.
object  An object from class "zpssR" (output of `zipfpssFit` function).
x      An object from class "zpssR" (output of `zipfpssFit` function).

Details

The argument `data` is a two column matrix with the first column containing the observations and the second column containing their frequencies.

References


See Also

`getInitialValues`.

Examples

data <- rzipfpss(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(as.character(data[,1]))
data[,2] <- as.numeric(as.character(data[,2]))
obj <- zipfpssFit(data, init_alpha = 1.5, init_lambda = 1.5)
Index

AIC.moezipfR (moezipfFit), 5
AIC.zi_zipfpssR (zi_zipfpssFit), 24
AIC.zipfpeR (zipfpeFit), 11
AIC.zipfPolyR (zipfPolylogFit), 16
AIC.zipfpssR (zipfpssFit), 18

BIC.moezipfR (moezipfFit), 5
BIC.zi_zipfpssR (zi_zipfpssFit), 24
BIC.zipfpeR (zipfpeFit), 11
BIC.zipfPolyR (zipfPolylogFit), 16
BIC.zipfpssR (zipfpssFit), 18

coef.moezipfR (moezipfFit), 5
coef.zi_zipfpssR (zi_zipfpssFit), 24
coef.zipfpeR (zipfpeFit), 11
coef.zipfPolyR (zipfPolylogFit), 16
coef.zipfpssR (zipfpssFit), 18

d_zi_zipfpss (zi_zipfpss), 23
dmoezipf (moezipf), 4
dzipfpe (zipfpe), 9
dzipfPolylog (zipfPolylog), 15
dzipfpss (zipfpss), 17

fitted.moezipfR (moezipfFit), 5
fitted.zi_zipfpssR (zi_zipfpssFit), 24
fitted.zipfpeR (zipfpeFit), 11
fitted.zipfPolyR (zipfPolylogFit), 16
fitted.zipfpssR (zipfpssFit), 18

getInitialValues, 2, 7, 12, 20, 25

logLik.moezipfR (moezipfFit), 5
logLik.zi_zipfpssR (zi_zipfpssFit), 24
logLik.zipfpeR (zipfpeFit), 11
logLik.zipfPolyR (zipfPolylogFit), 16
logLik.zipfpssR (zipfpssFit), 18

moezipf, 4
moezipfFit, 5
moezipfMean, 7, 8, 9
moezipfMoments, 8, 9
moezipfVariance, 9

optim, 6, 7, 12, 17, 19, 25

plot.moezipfR (moezipfFit), 5
plot.zi_zipfpssR (zi_zipfpssFit), 24
plot.zipfpeR (zipfpeFit), 11
plot.zipfPolyR (zipfPolylogFit), 16
plot.zipfpssR (zipfpssFit), 18
pmoezipf (moezipf), 4
print.moezipfR (moezipfFit), 5
print.zi_zipfpssR (zi_zipfpssFit), 24
print.zipfpeR (zipfpeFit), 11
print.zipfPolyR (zipfPolylogFit), 16
print.zipfpssR (zipfpssFit), 18
pzipfpe (zipfpe), 9
pzipfpss (zipfpss), 17

qmoezipf (moezipf), 4
qzipfpe (zipfpe), 9
qzipfpss (zipfpss), 17

residuals.moezipfR (moezipfFit), 5
residuals.zi_zipfpssR (zi_zipfpssFit), 24
residuals.zipfpeR (zipfpeFit), 11
residuals.zipfPolyR (zipfPolylogFit), 16
residuals.zipfpssR (zipfpssFit), 18
rmoezipf (moezipf), 4
rzipfpe (zipfpe), 9
rzipfpss (zipfpss), 17

summary.moezipfR (moezipfFit), 5
summary.zi_zipfpssR (zi_zipfpssFit), 24
summary.zipfpeR (zipfpeFit), 11
summary.zipfPolyR (zipfPolylogFit), 16
summary.zipfpssR (zipfpssFit), 18
zi_zipfpss, 23
zi_zipfpssFit, 24
INDEX

zipfpe, 9
zipfpeFit, 11
zipfpeMean, 13, 14, 15
zipfpeMoments, 14, 15
zipfpeVariance, 14
zipfPolylog, 15
zipfPolylogFit, 16
zipfpss, 17
zipfpssFit, 18
zipfpssMean, 20
zipfpssMoments, 21
zipfpssVariance, 22