Package ‘zipfextR’

March 7, 2018

Type Package

Title Zipf Extended Distributions

Version 0.7.0

Author Ariel Duarte-López and Marta Pérez-Casany

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

Description Implementation of three extensions of the Zipf distribution: the Marshall-Olkin
Extended Zipf (MOEZipf), the Zipf-Poisson Extreme (Zipf-PE) and the
Zipf-Poisson Stopped Sum (Zipf-PSS) 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)

Encoding UTF-8

LazyData true

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

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

RoxygenNote 6.0.1

Suggests testthat

R topics documented:

`getInitialValues` *Calculates initial values for the α and β parameters.*

Description

The selection of robust initial values allows to reduce the number of iterations which in turn, reduces the computation time. The initial values proposed by this function are computed using the empirical absolute frequencies of the first two consecutive values (i.e. 0 and 1, or, 1 and 2 when the distribution does not contains the zero on its support). In the case where one of these the two positive integer values does not appear in the data set, the default values are set to be equal to the minimum values in the support of the parameters of the model.

Usage

```
getInitialValues(data, model = "zipf")
```

Arguments

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

Details

The argument `data` is a two column matrix with the first column containing the observations and the second column containing their frequencies. On the other hand, the argument `model` refers to all the models implemented in the package. The possible values are *moezipf*, *zipfpe*, *zipfpss* or its zero truncated version *zt_zipfpss*.

For those models that allow to obtain the Zipf distribution as a particular case, the value of the α parameter is set to be equal to:

$$\alpha_0 = log_2(\frac{f_a(1)}{f_a(2)})$$

where $f_a(1)$ and $f_a(2)$ are the absolute frequencies of one and two in the sample. On the other hand, the second parameter of the model is set to be equal to the value that give place to have the same probabilities as the Zipf distribution.

In the particular case of the Zipf-PSS which does not have included the Zipf distribution the initial value proposed for the λ parameter is:

$$\lambda_0 = -log(f_r(o)),$$

where $f_r(0)$ is the relative frequency associated to the zero probability. The value of the α_0 is obtaining by means of an optimization process and it is set to be equal to the solution of the following equation:

$$\alpha_0 = \zeta^{-1}(\lambda_0 * f_a(0)/f_a(1)),$$

where $f_a(0)$ and $f_a(1)$ are the absolute frequencies associated to the values 0 and 1 respectively.

Remark: The user is in charge of considerer to use or not the proposed initial values. In any case it will change the estimation provided for estimating the likelihood parameters of the distribution.

Value

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

References

Güney, Y., Tuaç, Y., & Arslan, O. (2016). Marshall–Olkin distribution: parameter estimation and application to cancer data. Journal of Applied Statistics, 1-13.

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)
```

`moezipf` *The Marshal-Olkin Extended Zipf Distribution (MOEZipf).*

Description

Probability mass function, cumulative distribution function, quantile function and random number generation for the MOEZipf distribution with parameters α and β. 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 α parameter ($\alpha > 1$).
`beta`	Value of the β 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 α and β is computed as follows:

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

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

$$S(x) = \frac{\beta\,\zeta(\alpha,x+1)}{\zeta(\alpha) - \bar{\beta}\,\zeta(\alpha,x+1)},\ x = 1,2,..$$

The quantile of the MOEZipf(α,β) distribution of a given probability value p is equal to the quantile of the Zipf(α) distribution at the value:

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

The quantiles of the Zipf(α) 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

Casellas, A. (2013) *La distribució Zipf Estesa segons la transformació Marshall-Olkin*. Universitat Politécnica de Catalunya.

Devroye L. (1986) Non-Uniform Random Variate Generation. Springer, New York, NY.

Duarte-López, A., Prat-Pérez, A., & Pérez-Casany, M. (2015). *Using the Marshall-Olkin Extended Zipf Distribution in Graph Generation*. European Conference on Parallel Processing, pp. 493-502, Springer International Publishing.

Pérez-Casany, M. and Casellas, A. (2013) *Marshall-Olkin Extended Zipf Distribution*. arXiv preprint arXiv:1304.4540.

Young, D. S. (2010). *Tolerance: an R package for estimating tolerance intervals*. Journal of Statistical Software, 36(5), 1-39.

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)
```

`moezipfFit` *MOEZipf 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 MOEZipf distribution by means of the maximum likelihood method. The input data should be provided as a frequency matrix.

Usage

```
moezipfFit(data, init_alpha, init_beta, 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 α parameter ($\alpha > 1$).
`init_beta`	Initial value of β 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. $N = \sum_{i=1}^{m} x_i f_a(x_i)$.

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

```
data <- rmoezipf(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(levels(data[,1])[data[,1]])
initValues <- getInitialValues(data, model='moezipf')
obj <- moezipfFit(data, init_alpha = initValues$init_alpha, init_beta = initValues$init_p2)
```

`moezipfMean` *Expected value.*

Description

Computes the expected value of the MOEZipf distribution for given values of parameters α and β.

Usage

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

Arguments

`alpha`	Value of the α parameter ($\alpha > 2$).
`beta`	Value of the β parameter ($\beta > 0$).
`tolerance`	Tolerance used in the calculations (default = 10^{-4}).

Details

The mean of the distribution only exists for α 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

```
moezipfMean(2.5, 1.3)
moezipfMean(2.5, 1.3, 10^(-3))
```

`moezipfMoments` *Distribution Moments.*

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

```
moezipfMoments(k, alpha, beta, tolerance = 10^(-4))
```

Arguments

`k`	Order of the moment to compute.
`alpha`	Value of the α parameter ($\alpha > k + 1$).
`beta`	Value of the β 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

```
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 α and β.

Usage

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

Arguments

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

Details

The variance of the distribution only exists for α 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 α and β. 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 α parameter ($\alpha > 1$).
`beta`	Value of the β 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 α and β at a positive integer value x is computed as follows:

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

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

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

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

The quantile of the Zipf-PE(α, β) distribution of a given probability value p is equal to the quantile of the Zipf(α) distribution at the value:

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

The quantiles of the Zipf(α) 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

Young, D. S. (2010). *Tolerance: an R package for estimating tolerance intervals*. Journal of Statistical Software, 36(5), 1-39.

Examples

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

---

zipfpeFit *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, init_beta, 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, ...)

## S3 method for class 'zipfpeR'
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 α parameter ($\alpha > 1$).
`init_beta`	Initial value of β 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:

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

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. $N = \sum_{i=1}^{m} x_i f_a(x_i)$.

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

```
data <- rzipfpe(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(levels(data[,1])[data[,1]])
initValues <- getInitialValues(data, model='zipfpe')
obj <- zipfpeFit(data, init_alpha = initValues$init_alpha, init_beta = initValues$init_p2)
```

`zipfpeMean` *Expected value of the Zipf-PE distribution.*

Description

Computes the expected value of the Zipf-PE distribution for given values of parameters α and β.

Usage

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

Arguments

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

Details

The mean of the distribution only exists for α 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

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

`zipfpeMoments` *Distribution Moments.*

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 zipfpeMean function.

Usage

```
zipfpeMoments(k, alpha, beta, tolerance = 10^(-4))
```

Arguments

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

Details

The k-th moment of the Zipf-PE distribution is finite for α 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

```
zipfpeMoments(3, 4.5, 1.3)
zipfpeMoments(3, 4.5, 1.3,  1*10^(-3))
```

`zipfpeVariance` *Variance of the Zipf-PE distribution.*

Description

Computes the variance of the Zipf-PE distribution for given values of α and β.

Usage

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

Arguments

`alpha`	Value of the α parameter ($\alpha > 3$).
`beta`	Value of the β parameter ($\beta \in (-\infty, +\infty)$).
`tolerance`	Tolerance used in the calculations. (default = 10^{-4})

Details

The variance of the distribution only exists for α 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)
```

---

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 α and λ. 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 α parameter ($\alpha > 1$).
`lambda`	Value of the λ 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 λ 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

Panjer, H. H. (1981). Recursive evaluation of a family of compound distributions. ASTIN Bulletin: The Journal of the IAA, 12(1), 22-26.

Sundt, B., & Jewell, W. S. (1981). Further results on recursive evaluation of compound distributions. ASTIN Bulletin: The Journal of the IAA, 12(1), 27-39.

`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

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

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

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

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

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

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

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

## 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 α parameter ($\alpha > 1$).
`init_lambda`	Initial value of λ 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.

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

Value

Returns a *zpssR* 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

Panjer, H. H. (1981). Recursive evaluation of a family of compound distributions. ASTIN Bulletin: The Journal of the IAA, 12(1), 22-26.

Sundt, B., & Jewell, W. S. (1981). Further results on recursive evaluation of compound distributions. ASTIN Bulletin: The Journal of the IAA, 12(1), 27-39.

See Also

`getInitialValues`.

Examples

```
data <- rzipfpss(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(levels(data[,1])[data[,1]])
initValues <- getInitialValues(data, model='zipfpss')
obj <- zipfpssFit(data, init_alpha = initValues$init_alpha, init_lambda = initValues$init_p2)
```

`zipfpssMean` *Expected value of the Zipf-PSS distribution.*

Description

Computes the expected value of the Zipf-PSS distribution for given values of parameters α and λ.

Usage

```
zipfpssMean(alpha, lambda, isTruncated = FALSE)
```

Arguments

`alpha`	Value of the α parameter ($\alpha > 2$).
`lambda`	Value of the λ 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 α 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:

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

Particularlly, 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

Sarabia Alegría, J. M., Gómez Déniz, E. M. I. L. I. O., & Vázquez Polo, F. (2007). Estadística actuarial: teoría y aplicaciones. Pearson Prentice Hall.

Examples

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

`zipfpssMoments` *Distribution Moments.*

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 α parameter ($\alpha > k + 1$).
`lambda`	Value of the λ parameter ($\lambda \geq 0$).
`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 α 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

```
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 α and λ.

Usage

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

Arguments

`alpha`	Value of the α parameter ($\alpha > 3$).
`lambda`	Value of the λ parameter ($\lambda \geq 0$).
`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 α values strictly greater than 3. The value is obtained from the *law of total variance* that says that:

$$Var[Y] = E[N]\,Var[X] + E[X]^2\,Var[N],$$

where X follows a Zipf distribution with parameter α, and N follows a Poisson distribution with parameter λ. From where one has that:

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

Particularlly, 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

Sarabia Alegría, JM. and Gómez Déniz, E. and Vázquez Polo, F. Estadística actuarial: teoría y aplicaciones. Pearson Prentice Hall.

Examples

```
zipfpssVariance(4.5, 2.3)
zipfpssVariance(4.5, 2.3, TRUE)
```

Index