Package ‘zipfextR’

February 20, 2018

Type Package

Title Zipf Extended Distributions

Version 0.6.0

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. The two first extensions allow for top-concavity (top-convexity) while the third one only allows for concavity. In log-log scale, all the extensions maintain the linearity, associated with the Zipf model, in the tail. This can be appreciated by plotting the probabilities in log-log scale.

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:

moezipf 2
moezipfFit 4
moezipfMean 6
moezipfMoments 7
moezipfVariance 8
moezipf_getInitialValues 8

zipfpe 9
zipfpeFit 11
zipfpeMean 13
zipfpeMoments 14
zipfpeVariance 14
zipfpss 15
zipfpssFit 16
zipfpssMean 18
zipfpssMoments 19
zipfpssVariance 20

Index **21**

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

Description

Probability mass function, cumulative function, quantile function and random number generation for the MOEZipf distribution with parameters α and β.

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.

The random generator function applies the *quantile* function over *n* values from an Uniform distribution in the interval (0, 1,) in order to obtain the random values.

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, August). *Using the Marshall-Olkin Extended Zipf Distribution in Graph Generation*. In 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 a MOEZipf distribution by means of the maximum likelihood method. Note that 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 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 such that the first column of each row contains a count, while the corresponding second column contains its frequency.

The log-likelihood function is equalt 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 m is the number of different values in the sample, N is the sample size, i.e. $N = \sum_{i=1}^{m} x_i f_a(x_i)$, being $f_a(x_i)$ is the absolute frequency of x_i.

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

Value

Returns a *moezipfR* object composed by the maximum likelihood parameter estimations, their standard deviation, their confidence intervals and the value of the log-likelihood at the maximum likelihood estimator.

See Also

`moezipf_getInitialValues`.

Examples

```
data <- rmoezipf(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(data[,1])
initValues <- moezipf_getInitialValues(data)
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 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 expected value of the MOEZipf distribution only exists for α values strictly greater than 2. In this case, if Y is a random variable that follows a MOEZipf distribution with parameters α and β, the expected value is equal to:

$$E(Y) = \sum_{x=1}^{\infty} \frac{\beta\zeta(\alpha)x^{-\alpha+1}}{[\zeta(\alpha) - \bar{\beta}\zeta(\alpha, x)][\zeta(\alpha) - \bar{\beta}\zeta(\alpha, x+1)]}, \, \alpha > 2, \, \beta > 0$$

The mean is computed calculating the partial sums of the serie, and stopping when two consecutive partial sums differs 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. Note that the k-th moment exists if and only if $\alpha > k + 1$. When 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 of the MOEZipf distribution is finite for α values strictly greater than $k+1$. For a random variable Y that follows a MOEZipf distribution with parameters α and β, the k-th moment is equal to:

$$E(Y^k) = \sum_{x=1}^{\infty} \frac{\beta\zeta(\alpha)x^{-\alpha+k}}{[\zeta(\alpha) - \bar{\beta}\zeta(\alpha, x)][\zeta(\alpha) - \bar{\beta}\zeta(\alpha, x+1)]}\,, \alpha \geq k + 1\,, \beta > 0$$

The k-th moment is computed calculating the partial sums of the serie, and stopping when two consecutive partial sums differs 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. In this case, it is calculated as:

$$Var[Y] = E[Y^2] - (E[Y])^2$$

, where the first and the second moments are computed using the *moezipfMoments* function.

Value

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

Examples

```
moezipfVariance(3.5, 1.3)
```

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

Description

Initial values of the parameters useful to find the maximum likelihood estimators and required in the *moezipfFit* procedure. They may be computed using the empirical absolute frequencies of values one and two. The selection of robust initial values allows to reduce the number of iterations which in turn, reduces the computation time. In the case where one of the two first positive integer values does not appear in the data set, the default values are set to be equal to α = 1.0001 and β = 0.0001.

Usage

```
moezipf_getInitialValues(data)
```

Arguments

`data`	Matrix of count data.

Details

The argument `data` is a two column matrix such that the first column of each row contains a count, while the corresponding second column contains its frequency.

To obtain the initial value of α and β, it is assumed that the data come from a Zipf(α) distribution. The initial value for β is set to be equal to one, and the inital value for α, denoted by α_0, is obtained equating the ratio of the theoretical probabilities at one and two to the corresponding empirical ratio. this gives:

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

where f_1 and f_2 are the absolute frequencies of one and two in the sample.

Value

Returns the initial value for parameters α and β.

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(data[,1])
initials <- moezipf_getInitialValues(data)
```

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

Description

Probability mass function, cumulative function, quantile function and random number generation for the Zipf-PE distribution with parameters α and β.

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 ($\beta > 0$).
`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* of the Zipf-PE distribution at a positive integer value x with parameters α and β 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*, $F(x)$, at a given positive integer value x, is calcuted as:

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

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.

The random generator function applies the *quantile* function over *n* values from an Uniform distribution in the interval (0, 1,) in order to obtain the random values.

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 a Zipf-PE distribution by means of the maximum likelihood method.Note that 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 such that the first column of each row contains a count, while the corresponding second column contains its frequency.

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

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

Value

Returns an object composed by the maximum likelihood parameter estimations, their standard deviation, their confidence intervals and the value of the log-likelihood at the maximum likelihood estimations.

See Also

`moezipf_getInitialValues`.

Examples

```
data <- rzipfpe(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(data[,1])
obj <- zipfpeFit(data, 1.1, 0.1)
```

---

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 calculating the partial sums of the serie, and stopping when two consecutive partial sums differs 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. Note that the k-th moment exists if and only if $\alpha > k + 1$.

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 calculating the partial sums of the serie, and stopping when two consecutive partial sums differs 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. In this case, it is calculated as:

$$Var[Y] = E[Y^2] - (E[Y])^2,$$

where the first and the second moments are computed using the *zipfpeMoments* function.

Value

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

Examples

```
zipfpeVariance(3.5, 1.3)
```

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

Description

Probability mass function, cumulative function, quantile function and random number generation for the Zipf-PSS distribution with parameters α and λ.

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 ($\lambda \geq 0$).
`log, log.p`	Logical; if TRUE, probabilities p are given as log(p).
`isTruncated`	Logical; if TRUE, the 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.

References

Bjørn Sundt and William S Jewell. 1981. Further results on recursive evaluation of compound distributions. ASTIN Bulletin: The Journal of the IAA 12, 1 (1981), 27–39.

Harry H Panjer. 1981. Recursive evaluation of a family of compound distributions. Astin Bulletin 12, 01 (1981), 22–26.

`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 a Zipf-PSS distribution by means of the maximum likelihood method. Note that 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 ($\lambda \geq 0$).
`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 such that the first column of each row contains a count, while the corresponding second column contains its frequency.

The log-likelihood function is equalt 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, their standard deviation, their confidence intervals and the value of the log-likelihood at the maximum likelihood estimator.

References

Bjørn Sundt and William S Jewell. 1981. Further results on recursive evaluation of compound distributions. ASTIN Bulletin: The Journal of the IAA 12, 1 (1981), 27–39.

Harry H Panjer. 1981. Recursive evaluation of a family of compound distributions. Astin Bulletin 12, 01 (1981), 22–26.

See Also

`moezipf_getInitialValues`.

Examples

```
data <- rzipfpss(100, 2.5, 1.3)
data <- as.data.frame(table(data))
data[,1] <- as.numeric(data[,1])
obj <- zipfpssFit(data, 1.1, 0.1)
```

`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 ($\lambda \geq 0$).
`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 derived from $E[Y] = E[N]\,E[X]$ where E[X] is the mean value of the Zipf distribution and E[N] is the expected value of a Poisson one.

The resulting expression is set to be equal to:

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

Particularlly, if one is dealing with the zero-truncated version of the Zipf-PSS distribution. This values is calculated 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, JM. and Gómez Déniz, E. and Vázquez Polo, F. 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. Note that 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 calculating the partial sums of the serie, and stopping when two consecutive partial sums differs 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 derived from $Var[Y] = E[N]\,Var[X] + E[X]^2\,Var[N]$ where E[X] and E[N] are the expected value of the Zipf and the Poisson distributions respectively. In the same way the values of Var[X] and Var[N] stand for the variances of the Zipf and the Poisson distributions. The resulting expression is set to be equal to:

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

Particularlly, the variance of the zero-truncated version of the Zipf-PSS distribution is calculated 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

AIC.moezipfR *(moezipfFit)*, 4
AIC.zipfpeR *(zipfpeFit)*, 11
AIC.zipfpssR *(zipfpssFit)*, 16

BIC.moezipfR *(moezipfFit)*, 4
BIC.zipfpeR *(zipfpeFit)*, 11
BIC.zipfpssR *(zipfpssFit)*, 16

coef.moezipfR *(moezipfFit)*, 4
coef.zipfpeR *(zipfpeFit)*, 11
coef.zipfpssR *(zipfpssFit)*, 16

dmoezipf *(moezipf)*, 2
dzipfpe *(zipfpe)*, 9
dzipfpss *(zipfpss)*, 15

fitted.moezipfR *(moezipfFit)*, 4
fitted.zipfpeR *(zipfpeFit)*, 11
fitted.zipfpssR *(zipfpssFit)*, 16

logLik.moezipfR *(moezipfFit)*, 4
logLik.zipfpeR *(zipfpeFit)*, 11
logLik.zipfpssR *(zipfpssFit)*, 16

moezipf, 2
moezipf_getInitialValues, *5*, 8, *12*, *18*
moezipfFit, 4
moezipfMean, 6, *7*
moezipfMoments, 7
moezipfVariance, 8

optim, *5*, *12*, *17*

plot.moezipfR *(moezipfFit)*, 4
plot.zipfpeR *(zipfpeFit)*, 11
plot.zipfpssR *(zipfpssFit)*, 16
pmoezipf *(moezipf)*, 2
print.moezipfR *(moezipfFit)*, 4
print.zipfpeR *(zipfpeFit)*, 11
print.zipfpssR *(zipfpssFit)*, 16
pzipfpe *(zipfpe)*, 9
pzipfpss *(zipfpss)*, 15

qmoezipf *(moezipf)*, 2
qzipfpe *(zipfpe)*, 9
qzipfpss *(zipfpss)*, 15

residuals.moezipfR *(moezipfFit)*, 4
residuals.zipfpeR *(zipfpeFit)*, 11
residuals.zipfpssR *(zipfpssFit)*, 16
rmoezipf *(moezipf)*, 2
rzipfpe *(zipfpe)*, 9
rzipfpss *(zipfpss)*, 15

summary.moezipfR *(moezipfFit)*, 4
summary.zipfpeR *(zipfpeFit)*, 11
summary.zipfpssR *(zipfpssFit)*, 16

zipfpe, 9
zipfpeFit, 11
zipfpeMean, 13
zipfpeMoments, 14
zipfpeVariance, 14
zipfpss, 15
zipfpssFit, 16
zipfpssMean, 18
zipfpssMoments, 19
zipfpssVariance, 20