Package 'moezipfR'

March 29, 2017

Type Package

Title Marshall-Olkin Extended Zipf

Version 1.0.2

Date 2017-03-22

Author Ariel Duarte-López <aduarte@ac.upc.edu>, Aina Casellas <ainacasellas@hotmail.com>, Marta Pérez-Casany <marta.perez@upc.edu>

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

Description Statistical utilities for the analysis of data by means of the Marshall-Olkin Extended Zipf distribution are presented. The distribution is a two-parameter extension of the widely used Zipf model. By plotting the probabilities in log-log scale, this two-parameter extension allows a concave as well as a convex behavior of the function at the beginning of the distribution, maintaining the linearity, associated to the Zipf model, in the tail.

License GPL-3

Depends R (>= 2.0.1)

RoxygenNote 5.0.1

Imports VGAM (>= 0.9.8), tolerance(>= 1.2.0)

Encoding UTF-8

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2017-03-29 14:06:01 UTC

R topics documented:

`moezipfR-package` *Marshall-Olkin Extended Zipf Distribution*

Description

Statistical utilities for the analysis of data by means of the Marshall-Olkin Extended Zipf distribution are presented. The distribution is a two-parameter extension of the widely used Zipf model. By plotting the probabilities in log-log scale, this two-parameter extension allows a concave as well as a convex behavior of the function at the beginning of the distribution, maintaining the linearity, associated to the Zipf model, in the tail.

Details

Package:	moezipfR
Type:	Package
Version:	1.0.2
Date:	2017-03-22
License:	GPL-3

Author(s)

Duarte-López, A., Casellas, A. and Pérez-Casany, M.

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

References

Casellas, A. (2013) *La distribució Zipf Estesa segons la transformació Marshall-Olkin*. Universitat Politécnica de Catalunya http://upcommons.upc.edu/handle/2099.1/18157

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.

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

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

---

`dmoezipf` *Density function.*

Description

Density function for the Marshall-Olkin Extended Zipf distribution with parameters α and β.

Usage

```
dmoezipf(x, alpha, beta, log = FALSE, show.plot = F)
```

Arguments

`x`	Vector of positive integer values.
`alpha`	Value of the α parameter ($\alpha > 1$).
`beta`	Value of the β parameter ($\beta > 0$).
`log`	Logical; if TRUE, probabilities p are given as log(p).
`show.plot`	Logical; if TRUE shows the plot of the distibution (default = FALSE).

Details

The probability mass function at a positive integer value x of the MOEZipf distribution with parameters α and β is computed as follows:

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

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

Value

The probability associated to each value in vector `x`.

Examples

```
dmoezipf(1:10, 2.5, 1.3)
dmoezipf(1:10, 2.5, 1.3, show.plot = TRUE)
```

`moezipfR.fit` *MOEZipf parameters estimation.*

Description

For a given count data set, 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.

Usage

```
moezipfR.fit(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.
`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 intervals (default 0.95).

`...`	Further arguments to the generic functions. In case of the function *moezipfR.fit* the extra arguments are passing to the optim function.
`object`	An object from class "moezipfR" (output of *moezipfR.fit* function).
`x`	An object from class "moezipfR" (output of *moezipfR.fit* function).

Details

The argument `data` is a matrix where, for each row, the first column contains a count, and the second column contains its corresponding frequency.

The log-likelihood function is computed by means of the following equation:

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

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

where N is the sample size $N = \sum_{i=1}^{m} x_i f_a(x_i)$, m is the number of different values x_i in the sample, and $f_a(x_i)$ 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 log-likelihood value.

See Also

`moezipfR.utils.getDataMatrix`, `moezipfR.utils.getInitialValues`.

Examples

```
data <- rmoezipf(100, 2.5, 1.3)
data <- moezipfR.utils.getDataMatrix(data)
obj <- moezipfR.fit(data, 1.001, 0.001)
```

`moezipfR.loglikelihood`

Log-likelihood.

Description

Computes the value of the log-likelihood function for a given data set and parameter values.

Usage

```
moezipfR.loglikelihood(data, alpha, beta)
```

Arguments

`data`	Matrix of count data.
`alpha`	Value of the α parameter ($\alpha > 1$).
`beta`	Value of the β parameter ($\beta > 0$).

Details

The argument `data` is a matrix where, for each row, the first column corresponds to a count, and the second column contains its corresponding frequency.

The log-likelihood function is computed by means of the following equation:

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

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

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

Value

The log-likelihood value.

Examples

```
data <- rmoezipf(100, 2.5, 1.3)
data <- moezipfR.utils.getDataMatrix(data)
moezipfR.loglikelihood(data, 2.5, 1.3)
```

`moezipfR.mean` *Expected value.*

Description

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

Usage

```
moezipfR.mean(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 computed as:

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

The mean is computed calculating the partial sums of the serie, and it stops when two consecutive partial sums differs less than the `tolerance` value. The last partial sum is returned.

Value

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

Examples

```
moezipfR.mean(2.5, 1.3)
moezipfR.mean(2.5, 1.3, 10^(-3))
```

`moezipfR.moments` *Distribution Moments.*

Description

General function to compute the k-th moment of the distribution, for any $k \geq 1$ 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 moezipfR.mean function.

Usage

```
moezipfR.moments(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 computed as:

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

The k-th moment is computed calculating the partial sums of the serie, and it stops when two consecutive partial sums differs less than the `tolerance` value. The last partial sum is returned.

Value

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

Examples

```
moezipfR.moments(3, 4.5, 1.3)
moezipfR.moments(3, 4.5, 1.3,  1*10^(-3))
```

`moezipfR.utils.getDataMatrix` *Convert a sample vector to a frequency matrix.*

Description

Converts a sequence of values into a matrix of frequencies.

Usage

```
moezipfR.utils.getDataMatrix(values)
```

Arguments

`values`	Vector of positive integer values.

Value

The matrix of frequencies associated to the vector `values`.

Examples

```
data <- rmoezipf(100, 2.5, 1.3)
moezipfR.utils.getDataMatrix(data)
```

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

Description

The initial value of the parameters are 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 equal to α = 1.0001 and β = 0.0001.

Usage

```
moezipfR.utils.getInitialValues(data)
```

Arguments

`data`	Matrix of count data.

Details

The argument `data` is a matrix where, for each row, the first column corresponds to a count, and the second column contains its corresponding frequency.

To obtain the initial value for α and β, one will assume that the data come from a Zipf(α) distribution. Thus, the initial value for β is set 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 empririal ratio. Thus,

$$\alpha_0 = log_2(\frac{f_1}{f_2})$$

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.

See Also

`moezipfR.utils.getDataMatrix`

Examples

```
data <- rmoezipf(100, 2.5, 1.3)
data <- moezipfR.utils.getDataMatrix(data)
initials <- moezipfR.utils.getInitialValues(data)
```

`moezipfR.var` *Variance.*

Description

Computes the variance of the MOEZipf distribution for given values of α and β.

Usage

```
moezipfR.var(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. It is calculated as:

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

Value

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

Examples

```
moezipfR.var(3.5, 1.3)
```

pmoezipf *Cumulative function.*

Description

Cumulative distribution function for the MOEZipf distribution with parameters α and β.

Usage

```
pmoezipf(x, alpha, beta, log.p = FALSE, lower.tail = TRUE, show.plot = F)
```

Arguments

x	Vector of positive values.
alpha	Value of the α parameter ($\alpha > 1$).
beta	Value of the β parameter ($\beta > 0$).
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, $P[X > x]$.
show.plot	Logical; if TRUE shows the plot of the distibution (default = FALSE).

Details

The cumulative distribution function, $F(x)$, at a given positive real value x, is calcuted from the survival function $S(x)$ as:

$$F(x) = 1 - S(x),$$

the survival function $S(x)$ is equal to:

$$S(x) = \frac{\beta\zeta(\alpha, x+1)}{\zeta(\alpha) - \bar{\beta}\zeta(\alpha, x+1)}, \forall x > 0$$

Value

The cumulative probability of each value in vector x.

See Also

smoezipf for the survival probability function.

Examples

```
pmoezipf(1:10, 2.5, 1.3)
pmoezipf(1:10, 2.5, 1.3, show.plot = TRUE)
```

`qmoezipf` *Quantile function of the distribution.*

Description

Computes the inverse cumulative function for a given vector of probabilities p. It requires the `qzipfman` function implemented into the `tolerance` package refered below.

Usage

```
qmoezipf(p, alpha, beta, log.p = FALSE, lower.tail = TRUE)
```

Arguments

`p`	Vector of probabilities.
`alpha`	Value of the α parameter ($\alpha > 1$).
`beta`	Value of the β parameter ($\beta > 0$).
`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, $P[X > x]$.

Details

The quantiles of a MOEZipf distribution for a given probability vector p, are obtained by computing the quantiles associated to a Zipf distribution with the same parameter α, and probability vector equal to:

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

Value

Quantiles associated to a given probability vector p.

References

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

Examples

```
qmoezipf(0.56, 2.5, 1.3)
```

rmoezipf *Random number generator.*

Description

Generates random numbers from a MOEZipf distribution with fixed parameters α and β.

Usage

```
rmoezipf(n, alpha, beta)
```

Arguments

n	Number of random numbers to return.
alpha	Value of the α parameter ($\alpha > 1$).
beta	Value of the β parameter ($\beta > 0$).

Value

Vector containing the n generated numbers.

References

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

Examples

```
rmoezipf(10, 2.5, 1.3)
```

smoezipf *Survival function.*

Description

Survival function for the MOEZipf with parameters α, β.

Usage

```
smoezipf(x, alpha, beta, show.plot = F)
```

Arguments

x	Vector of positive integer values.
alpha	Value of the α parameter ($\alpha > 1$).
beta	Value of the β parameter ($\beta > 0$).
show.plot	logical; if TRUE shows the plot of the distibution (default = FALSE).

Details

The survival function at value x is computed as follows:

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

Value

The survival probability value associated to each component in vector `x`.

Examples

```
smoezipf(1:10, 2.5, 1.3)
smoezipf(1:10, 2.5, 1.3, show.plot = TRUE)
```

Index