Package: polyaAeppli (via r-universe)

September 2, 2024

Type Package

Title Implementation of the Polya-Aeppli Distribution

Version 2.0.2

Depends R (>= 3.0.0)

Date 2022-04-21

Author Conrad Burden

Maintainer Conrad Burden <conrad.burden@anu.edu.au>

Description Functions for evaluating the mass density, cumulative distribution function, quantile function and random variate generation for the Polya-Aeppli distribution, also known as the geometric compound Poisson distribution. More information on the implementation can be found at Conrad J. Burden (2014) <arXiv:1406.2780>.

NeedsCompilation no

Date/Publication 2022-04-21 11:10:04 UTC

Repository https://cjb105.r-universe.dev

RemoteUrl https://github.com/cran/polyaAeppli

RemoteRef HEAD

RemoteSha 395b5da1a54bf1ac130d25284aa792e802cf0580

Contents

polyaAeppli-package 2
PolyaAeppli 3

Index **6**

polyaAeppli-package *Implementation of the Polya-Aeppli Distribution*

Description

Functions for evaluating the mass density, cumulative distribution function, quantile function and random variate generation for the Polya-Aeppli distribution, also known as the geometric compound Poisson distribution.
More information on the implementation of **polyaAeppli** can be found at Conrad J. Burden (2014) <arXiv:1406.2780>.

Details

Package:	polyaAeppli
Type:	Package
Version:	2.0.2
Depends:	R (>= 3.0.0)
Date:	2020-04-21
License:	GPL(>=2)

Consistent with the conventions used in R package stats, this implementation of the Polya-Aeppli distribution comprises the four functions

```
dPolyaAeppli(x, lambda, prob, log = FALSE)
pPolyaAeppli(q, lambda, prob, lower.tail = TRUE, log.p = FALSE)
qPolyaAeppli(p, lambda, prob, lower.tail = TRUE, log.p = FALSE)
rPolyaAeppli(n, lambda, prob)
```

Author(s)

Conrad Burden

Maintainer: conrad.burden@anu.edu.au

References

Johnson NL, Kotz S, Kemp AW (1992). *Univariate Discrete Distributions.* 2nd edition. Wiley, New York.

Nuel G (2008). *Cumulative distribution function of a geometeric Poisson distribution.* Journal of Statistical Computation and Simulation, **78**(3), 385-394.

Examples

```
lambda <- 8
prob <- 0.2
## Plot histogram of random sample
PAsample <- rPolyaAeppli(10000, lambda, prob)
maxPA <- max(PAsample)
hist(PAsample, breaks=(0:(maxPA + 1)) - 0.5, freq=FALSE,
    xlab = "x", ylab = expression(P[X](x)), main="", border="blue")
## Add plot of density function
x <- 0:maxPA
points(x, dPolyaAeppli(x, lambda, prob), type="h", lwd=2)

lambda <- 4000
prob <- 0.005
qq <- 0:10000
## Plot log of the extreme lower tail p-value
log.pp <- pPolyaAeppli(qq, lambda, prob, log.p=TRUE)
plot(qq, log.pp, type = "l", ylim=c(-lambda,0),
      xlab = "x", ylab = expression("log Pr(X " <= "x)"))
## Plot log of the extreme upper tail p-value
log.1minuspp <- pPolyaAeppli(qq, lambda, prob, log.p=TRUE, lower.tail=FALSE)
points(qq, log.1minuspp, type = "l", col = "red")
legend("topright", c("lower tail", "upper tail"),
    col=c("black", "red"), lty=1, bg="white")
```

PolyaAeppli *Polya-Aeppli*

Description

Density, distribution function, quantile function and random generation for the Polya-Aeppli distribution with parameters lambda and prob.

Usage

```
dPolyaAeppli(x, lambda, prob, log = FALSE)
pPolyaAeppli(q, lambda, prob, lower.tail = TRUE, log.p = FALSE)
qPolyaAeppli(p, lambda, prob, lower.tail = TRUE, log.p = FALSE)
rPolyaAeppli(n, lambda, prob)
```

Arguments

x	vector of quantiles
q	vector of quantiles
p	vector of probabilities
n	number of random variables to return
lambda	a vector of non-negative Poisson parameters

prob	a vector of geometric parameters between 0 and 1
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 $P[X > x]$

Details

A Polya-Aeppli, or geometric compound Poisson, random variable is the sum of a Poisson number of identically and independently distributed shifted geometric random variables. Its distribution (with lambda= λ, prob= p) has density

$$Prob(X = x) = e^(-\lambda)$$

for $x = 0$;

$$Prob(X = x) = e^(-\lambda) \sum_{n=1}^{y} (\lambda^n)/(n!) choose(y-1, n-1) p^(y-n)(1-p)^n$$

for $x = 1, 2, \ldots$.

If an element of x is not integer, the result of dPolyaAeppli is zero, with a warning.

The quantile is right continuous: qPolyaAeppli(p, lambda, prob) is the smallest integer x such that $P(X \leq x) \geq p$.

Setting lower.tail = FALSE enables much more precise results when the default, lower.tail = TRUE would return 1, see the example below.

Value

dPolyaAeppli gives the (log) density, pPolyaAepploi gives the (log) distribution function, qPolyaAeppli gives the quantile function, and rPolyaAeppli generates random deviates.

Invalid lambda or prob will terminate with an error message.

Author(s)

Conrad Burden

References

Johnson NL, Kotz S, Kemp AW (1992). *Univariate Discrete Distributions.* 2nd edition. Wiley, New York.

Nuel G (2008). *Cumulative distribution function of a geometeric Poisson distribution.* Journal of Statistical Computation and Simulation, **78**(3), 385-394.

Examples

```
lambda <- 8
prob <- 0.2
## Plot histogram of random sample
PAsample <- rPolyaAeppli(10000, lambda, prob)
maxPA <- max(PAsample)
```

```
hist(PAsample, breaks=(0:(maxPA + 1)) - 0.5, freq=FALSE,
   xlab = "x", ylab = expression(P[X](x)), main="", border="blue")
## Add plot of density function
x <- 0:maxPA
points(x, dPolyaAeppli(x, lambda, prob), type="h", lwd=2)

lambda <- 4000
prob <- 0.005
qq <- 0:10000
## Plot log of the extreme lower tail p-value
log.pp <- pPolyaAeppli(qq, lambda, prob, log.p=TRUE)
plot(qq, log.pp, type = "l", ylim=c(-lambda,0),
      xlab = "x", ylab = expression("log Pr(X " <= "x)"))
## Plot log of the extreme upper tail p-value
log.1minuspp <- pPolyaAeppli(qq, lambda, prob, log.p=TRUE, lower.tail=FALSE)
points(qq, log.1minuspp, type = "l", col = "red")
legend("topright", c("lower tail", "upper tail"),
    col=c("black", "red"), lty=1, bg="white")
```

Index

dPolyaAeppli (PolyaAeppli), 3

PolyaAeppli, 3
polyaAeppli (polyaAeppli-package), 2
polyaAeppli-package, 2
pPolyaAeppli (PolyaAeppli), 3

qPolyaAeppli (PolyaAeppli), 3

rPolyaAeppli (PolyaAeppli), 3