Package: flexCountReg (via r-universe)

October 16, 2026

Type Package

Title Estimation of a Variety of Count Regression Models

Version 0.1.1

Maintainer Jonathan Wood <jwood2@iastate.edu>

Description An implementation of multiple regression models for count data. These include various forms of the negative binomial (NB-1, NB-2, NB-P, generalized negative binomial, etc.), Poisson-Lognormal, other compound Poisson distributions, the Generalized Waring model, etc. Information on the different forms of the negative binomial are described by Greene (2008) <doi:10.1016/j.econlet.2007.10.015>. For treatises on count models, see Cameron and Trivedi (2013) <doi:10.1017/CBO9781139013567> and Hilbe (2012) <doi:10.1017/CBO9780511973420>. For the implementation of under-reporting in count models, see Wood et al. (2016) <doi:10.1016/j.aap.2016.06.013>. For prediction methods in random parameter models, see Wood and Gayah (2025) <doi:10.1016/j.aap.2025.108147>. For estimating random parameters using maximum simulated likelihood, see Greene and Hill (2010) <doi:10.1108/S0731-9053(2010)26>; Gourieroux and Monfort (1996) <doi:10.1093/0198774753.001.0001>; or Hensher et al. (2015) <doi:10.1017/CBO9781316136232>.

Imports broom, cureplots, gsl, gt, knitr, MASS, maxLik, methods, modelr, purrr, randtoolbox, Rcpp (>= 1.0.12), rlang, sandwich, stats, dplyr, stringr, tibble, tidyr, truncnorm, utils

Depends R (>= 4.1.0)

LinkingTo Rcpp

SystemRequirements GNU make

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Collate 'Generalized-Waring.R' 'Lindley.R' 'RcppExports.R' 'cor2cov.R' 'corr_haltons.R' 'helpers.R' 'plind.R' 'ppoisGE.R' 'psichel.R' 'plindGamma.R' 'plindLnorm.R' 'ppoislogn.R' 'pinvgamma.R' 'pinvgaus.R' 'countreg.R' 'createFlexCountReg.R' 'tri.R' 'countregrp.R' 'cureplot.R' 'data.R' 'flexCountReg-Package.R' 'flexCountRegClass.R' 'genexp.R' 'globals.R' 'halton_dists.R' 'invgamma.R' 'metrics.R' 'mgf_lognormal.R' 'pcom.R' 'poisLind.re.R' 'poisWeib.R' 'predict_flexCountReg.R' 'regCompTest.R' 'regCompTable.R' 'renbLL.R' 'renb.R' 'summary_flexCountReg.R'

Suggests spelling, lamW, testthat (>= 3.0.0)

Config/testthat/edition 3

Language en-US

URL https://jwood-iastate.github.io/flexCountReg/

NeedsCompilation yes

Author Jonathan Wood [aut, cre] (ORCID: <https://orcid.org/0000-0003-0131-6384>), Guillermo Basulto-Elias [aut] (ORCID: <https://orcid.org/0000-0002-5205-2190>)

Config/pak/sysreqs make

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

Date/Publication 2026-01-19 18:20:20 UTC

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

RemoteRef HEAD

RemoteSha d164064f5a9bca254ec709f59f2601143ec2a5b5

Contents

COMDistribution 3
cor2cov 5
corr_haltons 5
countreg 7
countreg.rp 20
cureplot 22
dlindley 24
flexCountReg-class 25
Generalized-Waring 26
halton_dists 28
invgamma 29
mae 31
mgf_lognormal 32
myAIC 33
myBIC 34
NegativeBinomialLindley 35

poisLindRE 37
PoissonGeneralizedExponential 38
PoissonInverseGamma 41
PoissonInverseGaussian 42
PoissonLindley 46
PoissonLindleyLognormal 47
PoissonLognormal 49
PoissonWeibull 51
predict.flexCountReg 53
regCompTable 55
regCompTest 56
renb 58
rmse 60
SichelDistribution 61
summary.flexCountReg 62
tidy.flexCountReg 63
Triangular 63
washington_roads 65

Index **67**

`COMDistribution` *Conway-Maxwell-Poisson (COM) Distribution*

Description

These functions provide the density function, distribution function, quantile function, and random number generation for the Conway-Maxwell-Poisson (COM) Distribution

Usage

```
dcom(x, mu = NULL, lambda = 1, nu = 1, log = FALSE)

pcom(q, mu = NULL, lambda = 1, nu = 1, lower.tail = TRUE, log.p = FALSE)

qcom(p, mu = NULL, lambda = 1, nu = 1)

rcom(n, mu = NULL, lambda = 1, nu = 1)
```

Arguments

`x`	numeric value or a vector of values.
`mu`	optional. Numeric value or vector of mean values for the distribution (the values have to be greater than 0).
`lambda`	optional. Numeric value or vector of values for the rate parameter of the distribution (the values have to be greater than 0). If ‘mu‘ is provided, ‘lambda‘ is ignored.

nu	optional. Numeric value or vector of values for the decay parameter of the distribution ((the values have to be greater than 0).
log	logical; if TRUE, probabilities p are given as log(p).
q	quantile or a vector of quantiles.
lower.tail	logical; if TRUE, probabilities p are $P[X \leq x]$ otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).
p	probability or a vector of probabilities.
n	the number of random numbers to generate.

Details

dcom computes the density (PDF) of the COM Distribution.

pcom computes the CDF of the COM Distribution.

qcom computes the quantile function of the COM Distribution.

rcom generates random numbers from the COM Distribution.

The Probability Mass Function (PMF) for the Conway-Maxwell-Poisson distribution is:

$$f(x|\lambda,\nu) = \frac{\lambda^x}{(x!)^\nu Z(\lambda,\nu)}$$

Where λ and ν are distribution parameters with $\lambda > 0$ and $\nu > 0$, and $Z(\lambda,\nu)$ is the normalizing constant.

The normalizing constant is given by:

$$Z(\lambda,\nu) = \sum_{n=0}^{\infty} \frac{\lambda^n}{(n!)^\nu}$$

The mean and variance of the distribution are given by:

$$E[x] = \mu = \lambda \frac{\delta}{\delta\lambda} \log(Z(\lambda,\nu))$$

$$Var(x) = \lambda \frac{\delta}{\delta\lambda} \mu$$

When the mean value is given, the rate parameter (λ) is computed using the mean and the decay parameter (ν). This is useful to allow the calculation of the rate parameter when the mean is known (e.g., in regression))

Value

dcom gives the density, pcom gives the distribution function, qcom gives the quantile function, and rcom generates random deviates.

The length of the result is determined by n for rcom, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

Examples

```
dcom(1, mu=0.75, nu=3)
pcom(c(0,1,2,3,5,7,9,10), lambda=0.75, nu=0.75)
qcom(c(0.1,0.3,0.5,0.9,0.95), mu=0.75, nu=0.75)
rcom(30, mu=0.75, nu=0.5)
```

`cor2cov` *Generate a covariance matrix using a correlation matrix and vector of standard deviations*

Description

Generate a covariance matrix using a correlation matrix and vector of standard deviations

Usage

```
cor2cov(C, S)
```

Arguments

`C`	A correlation matrix.
`S`	A vector of standard deviations.

Value

A covariance matrix

Examples

```
C <- matrix(c(1,-0.3,0.7,-0.3,1,-0.2,0.7,-0.2,1), 3, 3)
S <- c(0.5, 2, 1.25)
cor2cov(C,S)
```

`corr_haltons` *Generate Correlated Random Variables Using Halton or Scrambled Halton Draws*

Description

This function generates `N` correlated random variables using Halton or scrambled Halton draws. The function supports normal and truncated normal distributions.

Usage

```
corr_haltons(
  means,
  cholesky = NULL,
  stdev = NULL,
  correlations = NULL,
  hdraws = NULL,
  ndraws = 500,
  scrambled = FALSE,
  dist = "normal",
  lower = -Inf,
  upper = Inf
)
```

Arguments

means	A numeric vector of means for each variable.
cholesky	A Cholesky decomposition matrix to introduce correlation.
stdev	A numeric vector of standard deviations for each variable. If provided, the function will use these values instead of the Cholesky decomposition matrix (must also provide a correlation matrix if providing standard deviations). Default is NULL.
correlations	A correlation matrix to introduce correlation. If provided, the function will use these values instead of the Cholesky decomposition matrix (must also provide standard deviations). Default is NULL.
hdraws	A matrix of Halton or scrambled Halton draws. If provided, the function will use these draws instead of generating new ones. Default is NULL.
ndraws	An integer specifying the number of values to simulate for each variable. Default is 500.
scrambled	A logical value indicating whether to use scrambled Halton draws. Default is FALSE.
dist	A character string specifying the distribution type. Options are "normal" and "truncated_normal". Default is "normal".
lower	A numeric value specifying the lower bound for truncated normal distribution. Default is -Inf.
upper	A numeric value specifying the upper bound for truncated normal distribution. Default is Inf.

Value

A matrix with N columns and ndraws rows containing the simulated values for the correlated random variables.

Examples

```
# Define mean, correlation, and standard deviations
means <- c(3, 2, 0.9)
sdevs <- c(0.25,1.5,0.8)
CORR <- matrix(c(1, -0.3, 0.5, -0.3, 1, -0.2, 0.5, -0.2, 1), 3, 3)

# Create the Cholesky decomposition matrix and set values for ndraws, etc.
ndraws <- 5000
scrambled <- TRUE
dist <- "normal"

# simulated the data
simulated_data <- corr_haltons(means, stdev=sdevs, correlations=CORR,
                               ndraws=ndraws, scrambled=scrambled,
                               dist=dist)

# look at the mean, standard deviation, and correlation of the simulated data
apply(simulated_data, 2, mean)
apply(simulated_data, 2, sd)
cor(simulated_data)

# providing a cholesky decomposition matrix
dist <- "normal"
cholesky <- chol(cor2cov(CORR, sdevs))
simulated_data <- corr_haltons(means, cholesky=cholesky, ndraws=ndraws,
                               scrambled=scrambled, dist=dist)
apply(simulated_data, 2, mean)
apply(simulated_data, 2, sd)
cor(simulated_data)

# Truncated normal
dist <- "truncated_normal"
lower <- 0
upper <- 30
simulated_data <- corr_haltons(means, cholesky=cholesky, ndraws=ndraws,
                               scrambled=scrambled, dist=dist,
                               lower=lower, upper=upper)
apply(simulated_data, 2, mean)
apply(simulated_data, 2, sd)
cor(simulated_data)
```

---

countreg *Count regression models*

Description

The purpose of this function is to estimate count regression models using maximum likelihood estimation (MLE) or Maximum Simulated Likelihood Estimation (MSLE). The function can estimate the following models:

- Poisson (Poisson)
- Negative Binomial 1 (NB1)
- Negative Binomial 2 (NB2)
- Negative Binomial P (NBP)
- Poisson-Lognormal (PLN)
- Poisson-Generalized-Exponential (PGE)
- Poisson-Inverse-Gaussian Type 1 (PIG1)
- Poisson-Inverse-Gaussian Type 2 (PIG2)
- Poisson-Inverse-Gamma (PIG)
- Poisson-Lindley (PL)
- Poisson-Lindley-Gamma (PLG), also known as the Negative Binomial-Lindley (NBL)
- Poisson-Lindley-Lognormal (PLL)
- Poisson-Weibull (PW)
- Sichel (SI)
- Generalized Waring (GW)
- Conway-Maxwell-Poisson (COM)

Usage

```
countreg(
  formula,
  data,
  family = "NB2",
  offset = NULL,
  weights = NULL,
  verbose = FALSE,
  dis_param_formula_1 = NULL,
  dis_param_formula_2 = NULL,
  underreport_formula = NULL,
  underreport_family = "logit",
  ndraws = 1500,
  method = "NM",
  max.iters = 1000,
  start.vals = NULL,
  stderr = "normal",
  bootstraps = NULL
)
```

Arguments

formula	a symbolic description of the model to be fitted.
data	a data frame containing the variables in the model.
family	the name of the distribution/model type to estimate. The default "NB2" is the standard negative binomial distribution with a log link. other options are listed below.

`offset` the name of a variable, or vector of variable names, in the data frame that should be used as an offset (i.e., included but forced to have a coefficient of 1). The normal method of setting an offset in the equation can also be used (overrides the offset option).

`weights` the name of a variable in the data frame that should be used as a frequency weight.

`verbose` an optional parameter. If 'TRUE', the function will print out the progress of the model fitting. Default is 'FALSE'.

`dis_param_formula_1`
a symbolic description of the model for the natural log of the dispersion parameter or first parameter of the count distribution used. Further details are provided below.

`dis_param_formula_2`
a symbolic description of the model for the second parameter of the count distribution used. Further details are provided below.

`underreport_formula`
an optional formula to estimate the underreporting for any of the count model options. The underreporting is estimated as a function (logit or probit) of the predictors in the model. For the model to be tractable, the independent variables cannot be the exact same as the count model. The default is 'NULL'.

`underreport_family`
the name of the distribution/model type to estimate the underreporting portion of the model when 'underreport_formula' is specified. The default is "logit" for a binary logistic regression model. The other option is "probit" for a probit model.

`ndraws` The number of Halton draws for integrating the distribution being compounded with the Poisson distribution when there is not a closed-form solution. Default is 1500. It is recommended to test different numbers of draws to determine if the model is stable (i.e., doesn't change or has minimal change as the number of draws changes within a reasonable range).

`method` Optimization method to be used for maximum likelihood estimation. See 'maxLik' documentation for options. The default is "NM" for the Nelder-Mead method.

`max.iters` Maximum number of iterations for the optimization method.

`start.vals` Optional vector of starting values for the optimization.

`stderr` Type of standard errors to use. The default is "normal". Other options include "boot" for bootstrapped standard errors, or "robust" for robust standard errors.

`bootstraps` Optional integer specifying the number of bootstrap samples to be used for estimating standard errors when 'stderr'= "boot". Note that this currently does not work when an offset variable is used.

Details

For the 'family' argument, the following options are available:

- "POISSON" for Poisson distribution with a log link.
- "NB1" for Negative Binomial 1 distribution with a log link.

- "NB2" for Negative Binomial 2 distribution with a log link (i.e., the standard negative binomial model).
- "NBP" for Negative Binomial P distribution with a log link.
- "PLN" for Poisson-Lognormal distribution with a log link.
- "PGE" for Poisson-Generalized-Exponential distribution with a log link.
- "PIG1" for Poisson-Inverse-Gaussian Type-1 distribution with a log link.
- "PIG2" for Poisson-Inverse-Gaussian Type-2 distribution with a log link.
- "PIG" for Poisson-Inverse-Gamma distribution with a log link.
- "PL" for Poisson-Lindley distribution with a log link.
- "PLG" for Poisson-Lindley-Gamma distribution with a log link.
- "PLL" for Poisson-Lindley-Lognormal distribution with a log link.
- "PW" for Poisson-Weibull distribution with a log link.
- "SI" for Sichel distribution with a log link.
- "GW" for Generalized Waring distribution with a log link.
- "COM" for Conway-Maxwell-Poisson (COM) distribution with a log link.

The ‘dis_param_formula_1‘ and ‘dis_param_formula_2‘ parameters are used to estimate the dispersion parameter or other parameters of the count distribution used. This leads to the distributions parameters being functions rather than constants in the model. For example, if the user wants to estimate the overdispersion parameter of the Negative Binomial 2 distribution as a function of the variable ‘x1‘ and ‘x2‘, the user would specify ‘dis_param_formula_1 = ~ x1 + x2‘. In the case of the Negative Binomial distributions, the model is known as a Generalized Negative Binomial model when the overdispersion parameter is specified as a function.

The function linking the distribution parameters to the predictors is:

$$Param = \exp((Intercept) + \sum \beta X)$$

The parameters for the different models are as follows:

For ‘dis_param_formula_1‘, the models are for the parameters:

- $ln(\alpha)$ for the Negative Binomial 1 model.
- $ln(\alpha)$ for the Negative Binomial 2 model.
- $ln(\alpha)$ for the Negative Binomial P model.
- $ln(\sigma)$ for the Poisson-Lognormal model.
- shape parameter for the Poisson-Generalized-Exponential model.
- $ln(\eta)$ for the Poisson-Inverse-Gaussian model.
- $ln(\eta)$ for the Poisson-Inverse-Gamma model.
- $ln(\theta)$ for the Poisson-Lindley model.
- $ln(\theta)$ for the Poisson-Lindley-Gamma model.
- $ln(\theta)$ for the Poisson-Lindley-Lognormal model.
- $ln(\alpha)$ for the Poisson-Weibull model.

- γ for the Sichel model.
- k for the Generalized Waring model.
- $ln(\nu)$ for the Conway-Maxwell-Poisson model.

For 'dis_param_formula_2', the models are for the parameters:

- Not Applicable for the Negative Binomial 1 model.
- Not Applicable for the Negative Binomial 2 model.
- p for the Negative Binomial P model.
- Not Applicable for the Poisson-Lognormal model.
- scale parameter for the Poisson-Generalized-Exponential model.
- Not Applicable for the Poisson-Inverse-Gaussian model.
- Not Applicable for the Poisson-Inverse-Gamma model.
- Not Applicable for the Poisson-Lindley model.
- $ln(\alpha)$ for the Poisson-Lindley-Gamma model.
- $ln(\sigma)$ for the Poisson-Lindley-Lognormal model.
- $ln(\sigma)$ for the Poisson-Weibull model.
- $ln(\sigma)$ for the Sichel model.
- $ln(\rho)$ for the Generalized Waring model.
- Not Applicable for the Conway-Maxwell-Poisson model.

The 'ndraws' parameter is used to estimate the distribution when there is not a closed-form solution. This uses Halton draws to integrate the distribution being compounded with the Poisson distribution. The default is 1500. The models this is applicable for include:

- Poisson-Lognormal
- Poisson-Generalized-Exponential
- Poisson-Lindley-Gamma (more efficient than using hypergeometric functions)
- Poisson-Lindley-Lognormal
- Poisson-Weibull

Value

An object of class 'countreg' which is a list with the following components:

- model: the fitted model object.
- data: the data frame used to fit the model.
- call: the matched call.
- formula: the formula used to fit the model.

Model Details

Poisson Model This implements the Poisson regression model using Maximum Likelihood Estimation, as opposed to the Iteratively Reweighted Least Squares (IRLS) method used in the 'glm' function.

The PMF and log-likelihood functions are:

$$P(Y = y) = \frac{e^{-\mu}\mu^y}{y!}$$

$$LL_{\text{Poisson}}(\beta) = \sum_{i=1}^{n} [-\mu_i + y_i \ln(\mu_i) - \ln(y_i!)]$$

The mean is:

$$\mu = exp(X\beta)$$

The variance is:

$$\text{Var}(Y) = \mu$$

Negative Binomial Models**

The NB-1, NB-2, and NB-P versions of the negative binomial distribution are based on Greene (2008). The details of each of these are provided below.

NB-1 Model The PMF and log-likelihood functions are:

$$P(Y = y) = \frac{\Gamma(y + \frac{\mu}{\alpha})}{y!\,\Gamma(\frac{\mu}{\alpha})} \left(\frac{\frac{\mu}{\alpha}}{\frac{\mu}{\alpha} + \mu}\right)^{\frac{\mu}{\alpha}} \left(\frac{\mu}{\frac{\mu}{\alpha} + \mu}\right)^{y}$$

$$LL_{\text{NB1}}(\beta, \alpha) = \sum_{i=1}^{n} \left[\ln \Gamma\left(y_i + \frac{\mu_i}{\alpha}\right) - \ln \Gamma\left(\frac{\mu_i}{\alpha}\right) - \ln y_i! + \frac{\mu_i}{\alpha} \ln\left(\frac{\frac{\mu_i}{\alpha}}{\frac{\mu_i}{\alpha} + \mu_i}\right) + y_i \ln\left(\frac{\mu_i}{\frac{\mu_i}{\alpha} + \mu_i}\right)\right]$$

The mean is:

$$\mu = exp(X\beta)$$

The variance is:

$$\text{Var}(Y) = \mu + \alpha\mu$$

NB-2 Model The PMF and log-likelihood functions are:

$$P(Y = y) = \frac{\Gamma(y + \alpha)}{y!\,\Gamma(\alpha)} \left(\frac{\alpha}{\alpha + \mu}\right)^{\alpha} \left(\frac{\mu}{\alpha + \mu}\right)^{y}$$

$$LL_{\text{NB2}} = \sum_{i=1}^{n} \left[\ln \Gamma(y_i + \alpha) - \ln \Gamma(\alpha) - \ln y_i! + \alpha \ln\left(\frac{\alpha}{\alpha + \mu_i}\right) + y_i \ln\left(\frac{\mu_i}{\alpha + \mu_i}\right)\right]$$

The mean is:

$$\mu = exp(X\beta)$$

The variance is:

$$\text{Var}(Y) = \mu + \alpha\mu^2$$

NB-P Model The PMF and log-likelihood functions are:

$$P(Y=y) = \frac{\Gamma(y+\frac{\mu^{2-p}}{\alpha})}{y!\,\Gamma(\frac{\mu^{2-p}}{\alpha})}\left(\frac{\frac{\mu^{2-p}}{\alpha}}{\frac{\mu^{2-p}}{\alpha}+\mu}\right)^{\frac{\mu^{2-p}}{\alpha}}\left(\frac{\mu}{\frac{\mu^{2-p}}{\alpha}+\mu}\right)^{y}$$

$$LL_{\text{NBP}}(\beta,\alpha,p) = \sum_{i=1}^{n}\left[\ln\Gamma\left(y_i+\frac{\mu_i^{2-p}}{\alpha}\right)-\ln\Gamma\left(\frac{\mu_i^{2-p}}{\alpha}\right)-\ln y_i!+\frac{\mu_i^{2-p}}{\alpha}\ln\left(\frac{\frac{\mu_i^{2-p}}{\alpha}}{\frac{\mu_i^{2-p}}{\alpha}+\mu_i}\right)+y_i\ln\left(\frac{\mu_i}{\frac{\mu_i^{2-p}}{\alpha}+\mu_i}\right)\right]$$

The mean is:

$$\mu = exp(X\beta)$$

The variance is:

$$\text{Var}(Y) = \mu+\alpha\mu^{P}$$

Poisson-Lognormal (PLN) Model The compound Probability Mass Function(PMF) for the Poisson-Lognormal distribution is:

$$f(y|\lambda,\sigma) = \int_0^{\infty}\frac{\lambda^y x^y e^{-\lambda x}}{y!}\frac{exp\left(-\frac{ln^2(x)}{2\sigma^2}\right)}{x\sigma\sqrt{2\pi}}dx$$

Where σ is a parameter for the lognormal distribution with the restriction $\sigma > 0$, and y is a non-negative integer.

The expected value of the distribution is:

$$E[y] = e^{X\beta+\sigma^2/2} = \mu e^{\sigma^2/2}$$

When 'ln.sigma.formula' is used, the parameter σ is modeled as:

$$ln(\sigma) = \beta_0+\beta_1x_1+\cdots+\beta_nx_n$$

Thus, the resulting value for the parameter σ is:

$$\sigma = e^{\beta_0+\beta_1x_1+\cdots+\beta_nx_n}$$

The t-statistics and p-values for the coefficients related to ln(sigma) are, by default, testing if the coefficients are different from a value of 0. This has little practical meaning given that they are coefficients for ln(sigma). They are not testing if the coefficients have statistical significance in terms of improvement over a Poisson model. The Likelihood-Ratio test results provided in the output provide a test comparing if the Poisson-Lognormal model provides a statistically significant improvement in model fit over the Poisson model.

Poisson Generalized-Exponential (PGE) Model The Generalized Exponential distribution can be written as a function with a shape parameter $\alpha > 0$ and scale parameter $\gamma > 0$. The distribution has strictly positive continuous values. The PDF of the distribution is:

$$f(x|\alpha,\gamma) = \frac{\alpha}{\gamma}\left(1-e^{-\frac{x}{\gamma}}\right)^{\alpha-1}e^{-\frac{x}{\gamma}}$$

Thus, the compound Probability Mass Function(PMF) for the PGE distribution is:

$$f(y|\lambda,\alpha,\beta) = \int_0^\infty \frac{\lambda^y x^y e^{-\lambda x}}{y!}\frac{\alpha}{\gamma}\left(1-e^{-\frac{x}{\gamma}}\right)^{\alpha-1} e^{-\frac{x}{\gamma}}dx$$

The expected value of the distribution is:

$$E[y] = \mu = \lambda\left(\frac{\psi(\alpha+1)-\psi(1)}{\gamma}\right)$$

Where $\psi(\cdot)$ is the digamma function.

The variance is:

$$\sigma^2 = \lambda\left(\frac{\psi(\alpha+1)-\psi(1)}{\gamma}\right) + \left(\frac{-\psi'(\alpha+1)+\psi'(1)}{\gamma^2}\right)\lambda^2$$

Where $\psi'(\cdot)$ is the trigamma function.

To ensure that $\mu = e^{X\beta}$, λ is replaced with:

$$\lambda = \frac{\gamma e^{X\beta}}{\psi(\alpha+1)-\psi(1)}$$

This results in:

$$f(y|\mu,\alpha,\beta) = \int_0^\infty \frac{\left(\frac{\gamma e^{X\beta}}{\psi(\alpha+1)-\psi(1)}\right)^y x^y e^{-\left(\frac{\gamma e^{X\beta}}{\psi(\alpha+1)-\psi(1)}\right)x}}{y!}\frac{\alpha}{\gamma}\left(1-e^{-\frac{x}{\gamma}}\right)^{\alpha-1} e^{-\frac{x}{\gamma}}dx$$

Halton draws are used to perform simulation over the lognormal distribution to solve the integral.

Poisson-Inverse-Gaussian Type 1 (PIG1) and Type 2 (PIG2) Models The Poisson-Inverse-Gaussian regression model is based on the Poisson-Inverse-Gaussian Distribution.

The expected value of the distribution in the regression utilizes a log-link function. Thus, the mean is:

$$\mu = e^{X\beta}$$

The variance function for the Type 1 distribution (which is the default) is:

$$\sigma^2 = \mu + \eta\mu$$

While the variance for the Type 2 distribution is:

$$\sigma^2 = \mu + \eta\mu^2$$

The parameter η is estimated as the natural logarithm transformed value, $\ln(\eta)$, to ensure that $\eta > 0$.

Poisson-Inverse-Gamma (PIG) Model The PDF of the distribution is:

$$f(x|\eta,\mu) = \frac{2\left(\mu\left(\frac{1}{\eta}+1\right)\right)^{\frac{x+\frac{1}{eta}+2}{2}}}{x!\Gamma\left(\frac{1}{\eta}+2\right)} K_{x-\frac{1}{\eta}-2}\left(2\sqrt{\mu\left(\frac{1}{\eta}+1\right)}\right)$$

Where η is a shape parameter with the restriction that $\eta > 0$, $\mu > 0$ is the mean value, y is a non-negative integer, and $K_i(z)$ is the modified Bessel function of the second kind. This formulation uses the mean directly.

The variance of the distribution is:

$$\sigma^2 = \mu + \eta\mu^2$$

Poisson-Lindley (PL) Model The Poisson-Lindley regression is based on a compound Poisson-Lindley distribution. It handles count outcomes with high levels of zero observations (or other high densities at low outcome values) that standard count regression methods, including the negative binomial, may struggle to adequately capture or model.

The compound Probability Mass Function(PMF) for the Poisson-Lindley (PL) distribution is:

$$f(y|\theta,\lambda) = \frac{\theta^2\lambda^y(\theta+\lambda+y+1)}{(\theta+1)(\theta+\lambda)^{y+2}}$$

Where θ and λ are distribution parameters with the restrictions that $\theta > 0$ and $\lambda > 0$, and y is a non-negative integer.

The expected value of the distribution is:

$$\mu = \frac{\lambda(\theta+2)}{\theta(\theta+1)}$$

If a log-link function is used, the mean is:

$$\mu = e^{X\beta} = \frac{\lambda(\theta+2)}{\theta(\theta+1)}$$

Thus, the parameter λ in the PL distribution when applied to regression analysis is:

$$\lambda = \frac{\mu\theta(\theta+1)}{\theta+2}$$

Using the replacement and simplifying results in:

$$f(y \mid \theta,\mu) = \frac{\theta^2(\mu\theta(\theta+1))^y(\theta^2(1+\mu)+\theta(2+\mu)+(\theta+2)(y+1))}{(\theta+1)(\theta+2)^{y+1}(\theta^2(1+\mu)+\theta(2+\mu))^{y+2}}$$

And

$$LL = 2\log(\theta)+y(\log(\mu)+\log(\theta)+\log(\theta+1))+\log(\theta^2(1+\mu)+\theta(2+\mu)+(\theta+2)(y+1))-\log(\theta+1)-(y+1)\log(\theta+2)-(y+2$$

The variance function is defined as:

$$\sigma^2 = \mu + \left(1 - \frac{2}{(\theta+2)^2}\right)\mu^2$$

It should be noted that the p-value for the parameter 'ln(theta)' in the model summary is testing if the parameter 'theta' is equal to a value of 1. This has no practical meaning. The Likelihood-Ratio (LR) test compares the Poisson-Lindley regression with a Poisson regression with the same independent variables. Thus, the PR test result indicates the statistical significance for the improvement in how

well the model fits the data over a Poisson regression. This indicates the statistical significance of the ‘theta‘ parameter.

Poisson-Lindley-Gamma (PLG) Model The Poisson-Lindley-Gamma regression is based on a compound Poisson-Lindley- Gamma distribution. Details of the distribution can be seen at dplindGamma.

The mean for the regression model is:

$$\mu = e^{X\beta}$$

The variance function is defined as:

$$\sigma^2 = \mu + \left(2\alpha + 1 - \frac{2(1+\alpha)}{(\theta+2)^2}\right)\mu^2$$

It should be noted that the p-value for the parameters ‘ln(theta)‘ and ‘ln(alpha)‘ in the model summary are testing if the parameter ‘theta‘ and ‘alpha‘ are equal to a value of 1.

Poisson-Lindley-Lognormal (PLL) Model The Poisson-Lindley-Lognormal regression is based on a compound Poisson- Lindley-Lognormal distribution. Details of the distribution can be seen at dplindLnorm.

The mean for the regression model is:

$$\mu = e^{X\beta}$$

The variance function is defined as:

$$\sigma^2 = \mu + \left(\frac{1 - \frac{2}{(\theta+2)^2}}{e^{\frac{\sigma^2}{2}}} + e^{\sigma^2} - 1\right)\mu^2$$

It should be noted that the p-value for the parameters ‘ln(theta)‘ and ‘ln(sigma)‘ in the model summary are testing if the parameter ‘theta‘ and ‘sigma‘ are equal to a value of 1.

Poisson-Weibull (PW) Model The Poisson-Weibull distribution uses the Weibull distribution as a mixing distribution for a Poisson process. It is useful for modeling overdispersed count data. The density function (probability mass function) for the Poisson-Weibull distribution is given by:

$$P(y|\lambda,\alpha,\sigma) = \int_0^\infty \frac{e^{-\lambda x}\lambda^y x^y}{y!}\left(\frac{\alpha}{\sigma}\right)\left(\frac{x}{\sigma}\right)^{\alpha-1} e^{-\left(\frac{x}{\sigma}\right)^\alpha} dx$$

where $f(x|\alpha,\sigma)$ is the PDF of the Weibull distribution and λ is the mean of the Poisson distribution.

For the Poisson-Weibull Regression model, the expected values is:

$$E[Y] = \lambda\sigma\Gamma\left(1 + \frac{1}{\alpha}\right)$$

Where λ is the mean of the Poisson distribution, α is the shape parameter, and σ is the scale parameter.

To ensure that the regression model predicts the mean value, the regression utilizes:

$$\mu = \exp X\gamma = \lambda\sigma\Gamma\left(1 + \frac{1}{\alpha}\right)$$

Where X is a matrix of independent variables and γ is a vector of coefficients.

This leads to:

$$\lambda = \frac{\mu}{\sigma\Gamma\left(1+\frac{1}{\alpha}\right)}$$

The variance for the Poisson-Weibull regression is:

$$V[Y] = \mu + \left(\frac{\Gamma\left(1+\frac{2}{\alpha}\right)}{\Gamma\left(1+\frac{1}{\alpha}\right)^2} - 1\right)\mu^2$$

Sichel (SI) Model The compound Probability Mass Function (PMF) for the Sichel distribution uses the formulation from Zhou et al. (2011) and Rigby et al. (2008):

$$f(y|\mu,\sigma,\gamma) = \frac{\left(\frac{\mu}{c}\right)^y K_{y+\gamma}(\alpha)}{K_\gamma(1/\sigma)y!(\alpha\sigma)^{y+\gamma}}$$

Where σ and γ are distribution parameters with $-\infty < \gamma < \infty$ and $\sigma > 0$, $c = \frac{K_{\gamma+1}(1/\sigma)}{K_\gamma(1/\sigma)}$, $\alpha^2 = \sigma^{-2} + 2\mu(c\sigma)^{-1}$, a mean value of μ, y is a non-negative integer, and $K_j(x)$ is a modified Bessel function of the third kind with order j and argument x.

The variance of the distribution is:

$$\sigma^2 = \mu + \left(\frac{2\sigma(\gamma+1)}{c} + \frac{1}{c^2} - 1\right)\mu^2$$

Generalized Waring (GW) Model The following are the versions of the PMF, mean, and variance used for the Generalized Waring model. This is adjusted from the typical formulation by replacing parameter k with μ

$$PMF = \frac{\Gamma(\alpha+y)\Gamma(k+y)\Gamma(\rho+k)\Gamma(\alpha+\rho)}{y!\Gamma(\alpha)\Gamma(k)\Gamma(\rho)\Gamma(\alpha+k+\rho+y)}$$

$$\mu = e^{X\beta} = \frac{\alpha k}{\rho-1}$$

$$\sigma^2 = \frac{\alpha k(\alpha+k+\rho-1)}{(\rho-1)^2(\rho-2)}$$

The distribution parameters are often considered to capture the randomness (parameter

$$\alpha$$

), proneness (parameter

k), and liability (parameter

$$\rho$$

) of the data.

If we use:

$$\alpha = \frac{\mu k}{\rho-1}$$

The PMF becomes:

$$PMF = \frac{\Gamma\left(\frac{\mu k}{\rho-1}+y\right)\Gamma(k+y)\Gamma(\rho+k)\Gamma\left(\frac{\mu k}{\rho-1}+\rho\right)}{y!\Gamma\left(\frac{\mu k}{\rho-1}\right)\Gamma(k)\Gamma(\rho)\Gamma\left(\frac{\mu k}{\rho-1}+k+\rho+y\right)}$$

This results in a regression model where:

$$\mu = e^{X\beta}$$

$$\sigma^2 = \frac{\mu k^2\left(\frac{\mu k}{\rho-1}+k+\rho-1\right)}{(\rho-1)^3(\rho-2)} = \left(\frac{k^3+\rho k^2-k^2}{(\rho-1)^3(\rho-2)}\right)\mu + \left(\frac{k^3}{(\rho-1)^4(\rho-2)}\right)\mu^2$$

Note that when

$$p = 1$$

or

$$p = 2$$

, the distribution is undefined.

Conway-Maxwell-Poisson (COM) Model The following is the the PMF for the COM model.

$$f(x|\lambda,\nu) = \frac{\lambda^x}{(x!)^\nu Z(\lambda,\nu)}$$

Where λ and ν are distribution parameters with $\lambda > 0$ and $\nu > 0$, and $Z(\lambda,\nu)$ is the normalizing constant.

The normalizing constant is given by:

$$Z(\lambda,\nu) = \sum_{n=0}^{\infty}\frac{\lambda^n}{(n!)^\nu}$$

The mean and variance are:

$$\mu = e^{X\beta} = \lambda\frac{\delta}{\delta\lambda}\log(Z(\lambda,\nu))$$

$$\sigma^2 = \lambda\frac{\delta}{\delta\lambda}\mu$$

Note that the COM distribution parameter λ is solved for using μ and ν, so the regression model provides direct predictions for the mean.

Underreporting Models for underreporting combine a binary probability model (logit or probit) with a count model. This is accomplished using a model for the probability of crashes being reported multiplied by the estimated mean for the count model, based on the observed data. This is discussed in Wood et. al. (2016), Pararai et. al., (2006), and Pararai et. al., (2010). The underreporting model is based on:

$$\mu_{true} = \mu_{observed} \cdot P(\text{event is reported})$$

This allows the inference of both the true event count and the probability of the event being reported as a function of independent variables.

References

Greene, W. (2008). Functional forms for the negative binomial model for count data. Economics Letters, 99(3), 585-590.

Pararai, M., Famoye, F., & Lee, C. (2006). Generalized Poisson regression model for underreported counts. Advances and applications in Statistics, 6(3), 305-322.

Pararai, M., Famoye, F., & Lee, C. (2010). Generalized Poisson-Poisson mixture model for misreported counts with an application to smoking data. Journal of Data Science, 8(4), 607-617.

Rigby, R. A., Stasinopoulos, D. M., & Akantziliotou, C. (2008). A framework for modelling overdispersed count data, including the Poisson-shifted generalized inverse Gaussian distribution. Computational Statistics & Data Analysis, 53(2), 381-393.

Wood, J.S., Eric T. Donnell, & Christopher J. Fariss. "A method to account for and estimate underreporting in crash frequency research." Accident Analysis & Prevention 95 (2016): 57-66.

Zou, Y., Lord, D., & Zhang, Y. (2012). Analyzing highly dispersed crash data using the Sichel generalized additive models for location, scale and shape.

Examples

```
# Load the Washington data
data("washington_roads")
washington_roads$AADT10kplus <- ifelse(washington_roads$AADT > 10000, 1, 0)
# Estimate an NB2 model with a dispersion parameter as a function of the
# variable `speed50` (i.e., generalized NB2), verbose output, and use the
# BFGS optimization method
nb2 <- countreg(Total_crashes ~ lnaadt + lnlength + speed50 + AADT10kplus,
               data = washington_roads, family = "NB2",
               dis_param_formula_1 = ~ speed50, verbose = TRUE,
               method='BFGS')
summary(nb2)


# Estimate a Poisson-Lognormal model (a low number of draws is used to speed
# up the estimation for examples - not recommended in practice)
pln <- countreg(Total_crashes ~ lnaadt + lnlength + speed50 + AADT10kplus,
              data = washington_roads, family = "PLN", ndraws=10)
summary(pln)

# Estimate an Poisson-Lognormal with underreporting (probit)
plogn_underreport <- countreg(Total_crashes ~ lnaadt + lnlength + speed50 +
               AADT10kplus,
              data = washington_roads, family = "NB2",
              underreport_formula = ~ speed50 + AADT10kplus,
              underreport_family = "probit")
summary(plogn_underreport)

# Estimate a Conway-Maxwell-Poisson model
com_model <- countreg(Total_crashes ~ lnaadt + lnlength + speed50 +
                      AADT10kplus,
                      data = washington_roads, family = "COM", method="BHHH")
summary(com_model)
```

```
#
```

`countreg.rp` *Random Parameters Count Regression Models*

Description

Random Parameters Count Regression Models

Usage

```
countreg.rp(
  formula,
  rpar_formula,
  data,
  family = "NB2",
  rpardists = NULL,
  dis_param_formula_1 = NULL,
  dis_param_formula_2 = NULL,
  het_mean_formula = NULL,
  het_var_formula = NULL,
  ndraws = 500,
  scrambled = FALSE,
  correlated = FALSE,
  panel_id = NULL,
  weights = NULL,
  offset = NULL,
  method = "BHHH",
  max.iters = 1000,
  start.vals = NULL,
  verbose = FALSE
)
```

Arguments

`formula`	an R formula. This formula should specify the outcome and the independent variables that have fixed parameters.
`rpar_formula`	a symbolic description of the model related specifically to the random parameters. This should not include an outcome variable. If the intercept is random, include it in this formula. If the intercept is fixed, include it in `formula` but not in `rpar_formula`.
`data`	a dataframe that has all of the variables in the `formula` and `rpar_formula`.
`family`	the name of the distribution/model type to estimate. Default is "NB2". Options include "Poisson", "NB1", "NB2", "NBP", "PIG", "Sichel", etc. (See `countreg` for full list).

rpardists	an optional named vector whose names are the random parameters and values the distribution. The distribution options include normal ("n"), lognormal ("ln"), triangular ("t"), uniform ("u"), and gamma ("g"). If not provided, normal is used.
dis_param_formula_1	a symbolic description of the model for the first parameter of the count distribution (e.g., ln(alpha) for NB2).
dis_param_formula_2	a symbolic description of the model for the second parameter of the count distribution (if applicable).
het_mean_formula	an optional symbolic description of the model for heterogeneity in the means of the random parameters.
het_var_formula	an optional symbolic description of the model for heterogeneity in the variances of the random parameters.
ndraws	the number of Halton draws to use for estimating the random parameters.
scrambled	if the Halton draws should be scrambled.
correlated	if the random parameters should be correlated. If TRUE, only normal distributions are used.
panel_id	an optional variable name (string) or vector defining the panel structure (repeated measures). If provided, the standard errors and likelihood are estimated accounting for the panel structure.
weights	variable name to be used as frequency weights.
offset	variable name to be used as an offset.
method	optimization method (e.g., "BHHH", "BFGS", "NM").
max.iters	maximum number of iterations.
start.vals	optional vector of starting values.
verbose	logical.

Value

An object of class ‘countreg‘ which is a list with the following components:

- model: the fitted model object.
- data: the data frame used to fit the model.
- call: the matched call.
- formula: the formula used to fit the model.

Examples

```
# Load data
data("washington_roads")
washington_roads$AADT10kplus <- ifelse(washington_roads$AADT > 10000, 1, 0)

# 1. Basic Random Parameters Negative Binomial (NB2)
```

```
rp_nb2 <- countreg.rp(Total_crashes ~ lnaadt + lnlength,
                      rpar_formula = ~ -1 + speed50,
                      data = washington_roads,
                      family = "NB2",
                      rpardists = c(speed50 = "n"),
                      ndraws = 100,
                      method = "BHHH")
summary(rp_nb2)

# 2. Random Parameters with Panel Structure (if 'site_id' exists)
# rp_panel <- countreg.rp(Total_crashes ~ -1 + lnaadt,
#                         rpar_formula = ~ speed50,
#                         data = washington_roads,
#                         panel_id = "site_id",
#                         family = "NB2",
#                         ndraws = 100)

# 3. Generalized Random Parameters Model with Heterogeneity
rp_gen <- countreg.rp(Total_crashes ~ lnaadt,
                      rpar_formula = ~ -1 + speed50,
                      dis_param_formula_1 = ~ lnlength,
                      het_mean_formula = ~ AADT10kplus,
                      data = washington_roads,
                      family = "NB2",
                      rpardists = c(speed50 = "n"),
                      ndraws = 100)
summary(rp_gen)

# 4. Random Parameters Poisson Model with panel specification
rp_poisson <- countreg.rp(Total_crashes ~ lnaadt,
                      rpar_formula = ~ -1 + speed50,
                      dis_param_formula_1 = ~ lnlength,
                      het_mean_formula = ~ AADT10kplus,
                      data = washington_roads,
                      family = "POISSON",
                      rpardists = c(speed50 = "n"),
                      ndraws = 100,
                      panel_id = "ID")
summary(rp_poisson)
```

`cureplot` *Cumulative Residuals (CURE) Plot for Count Models*

Description

This function generates a Cumulative Residuals (CURE) plot for count models, including those with random parameters, estimated using the flexCountReg package.

Usage

```
cureplot(
  model,
  data = NULL,
  indvar = NULL,
  method = "Simulated",
  n_resamples = 0,
  ...
)
```

Arguments

model	A model object estimated using this R package.
data	Optional dataframe. If not provided, the data used to fit the model will be used.
indvar	Optional independent variable name (character string). This is the continuous independent variable to plot the cumulative residuals against. If not provided, the plot will be against the predicted values.
method	Optional parameter to pass to the predict function. This is only used for random parameters models (e.g., "Simulated" or "Individual"). For further details, see predict.flexCountReg.
n_resamples	Number of resamples for potential resampling in the CURE plot confidence bands. Default is 0 (no bands).
...	Additional arguments passed to cure_plot.

Value

A CURE plot generated with **cureplots**.

Examples

```
## Example using a Negative Binomial model
data("washington_roads")
washington_roads$AADTover10k <- ifelse(washington_roads$AADT>10000,1,0)

nb_model <- countreg(Total_crashes ~ lnaadt + lnlength + speed50 +
                           ShouldWidth04 + AADTover10k,
                           data = washington_roads, family = 'nb2',
                           method = 'NM', max.iters = 500)

# 1. Plot against fitted values (default) with confidence bands
cureplot(nb_model, n_resamples = 20)

# 2. Plot against a specific covariate (e.g., lnlength)
cureplot(nb_model, indvar = "lnlength", n_resamples = 20)
```

`dlindley` *One-Parameter Lindley Distribution*

Description

Distribution function for the one-parameter Lindley distribution with parameter theta.

Usage

```
dlindley(x, theta = 1, log = FALSE)

plindley(q, theta = 1, lower.tail = TRUE, log.p = FALSE)

qlindley(p, theta = 1, lower.tail = TRUE, log.p = FALSE)

rlindley(n, theta = 1)
```

Arguments

`x`	a single value or vector of positive values.
`theta`	distribution parameter value. Default is 1.
`log, log.p`	logical; If TRUE, probabilities p are given as log(p). If FALSE, probabilities p are given directly. Default is FALSE.
`q`	a single value or vector of quantiles.
`lower.tail`	logical; If TRUE, (default), $P(X \leq x)$ are returned, otherwise $P(X > x)$ is returned. Default is TRUE.
`p`	a single value or vector of probabilities.
`n`	number of random values to generate.

Details

Probability density function (PDF)

$$f(x \mid \theta) = \frac{\theta^2}{(1+\theta)}(1+x)e^{-\theta x}$$

Cumulative distribution function (CDF)

$$F(x \mid \theta) = 1 - \left(1 + \frac{\theta x}{1+\theta}\right) e^{-\theta x}$$

Quantile function (Inverse CDF)

$$Q(p \mid \theta) = -1 - \frac{1}{\theta} - \frac{1}{\theta} W_{-1}\Big((1+\theta)(p-1)e^{-(1+\theta)}\Big)$$

where $W_{-1}()$ is the negative branch of the Lambert W function.

The moment generating function (MGF) is:

$$M_X(t) = \frac{\theta^2(\theta - t + 1)}{(\theta + 1)(\theta - t)^2}$$

The distribution mean and variance are:

$$\mu = \frac{\theta + 2}{\theta(1 + \theta)}$$

$$\sigma^2 = \frac{\mu}{\theta + 2}\left(\frac{6}{\theta} - 4\right) - \mu^2$$

Value

dlindley gives the density, plindley gives the distribution function, qlindley gives the quantile function, and rlindley generates random deviates.

The length of the result is determined by n for rlindley, and is the maximum of the lengths of the numerical arguments for the other functions.

Examples

```
x <- seq(0, 5, by = 0.1)
p <- seq(0.1, 0.9, by = 0.1)
q <- c(0.2, 3, 0.2)
dlindley(x, theta = 1.5)
dlindley(x, theta=0.5, log=TRUE)
plindley(q, theta = 1.5)
plindley(q, theta = 0.5, lower.tail = FALSE)
qlindley(p, theta = 1.5)
qlindley(p, theta = 0.5)

set.seed(123154)
rlindley(5, theta = 1.5)
rlindley(5, theta = 0.5)
```

---

flexCountReg-class *flexCountReg Class*

Description

A class to represent various objects created by the flexCountReg package.

Slots

`model` The fitted model object (can be any type).

`data` The data used for fitting the model (data frame).

`formula` The R formula used in the regression model (main formula, not including random parameters, etc.)

`call` The matched call (language).

`additional` Any additional information (list).

`Generalized-Waring` *Generalized Waring Distribution*

Description

These functions provide density, distribution function, quantile function, and random number generation for the Generalized Waring Distribution.

Usage

```
dgwar(y, mu, k, rho, log = FALSE)

pgwar(q, mu, k, rho, lower.tail = TRUE, log.p = FALSE)

qgwar(p, mu, k, rho)

rgwar(n, mu, k, rho)
```

Arguments

`y`	non-negative integer vector of count outcomes.
`mu`	numeric vector of means of the distribution.
`k`	non-negative numeric parameter of the distribution.
`rho`	non-negative numeric parameter of the distribution.
`log`	logical; if TRUE, probabilities p are given as log(p).
`q`	non-negative integer vector of quantiles.
`lower.tail`	logical; if TRUE, probabilities p are $P[X \leq x]$ otherwise, $P[X > x]$.
`log.p`	logical; if TRUE, probabilities p are given as log(p).
`p`	numeric vector of probabilities.
`n`	integer number of random numbers to generate.

Details

The Generalized Waring distribution is a 3-parameter count distribution that is used to model overdispersed count data.

`dgwar` computes the density (PMF) of the Generalized Waring Distribution.

`pgwar` computes the CDF of the Generalized Waring Distribution.

`qwaring` computes the quantile function of the Generalized Waring Distribution.

`rwaring` generates random numbers from the Generalized Waring Distribution.

The Probability Mass Function (PMF) for the Generalized Waring (GW) distribution is:

$$f(y|a_x, k, \rho) = \frac{\Gamma(a_x + \rho)\Gamma(k + \rho)\,(a_x)_y\,(k)_y}{y!\Gamma(\rho)\Gamma(a_x + k + \rho)(a_x + k + \rho)_y}$$

Where $(\alpha)_r = \frac{\Gamma(\alpha+r)}{\Gamma(\alpha)}$, and $a_x,\ k,\ \rho) > 0$.

The mean value is:

$$E[Y] = \frac{a_x K}{\rho - 1}$$

Thus, we can use:

$$a_x = \frac{\mu(\rho - 1)}{k}$$

This results in a regression model where:

$$\mu = e^{X\beta}$$

$$\sigma^2 = \mu\left(1 - \frac{1}{\alpha + \rho + 1}\right) + \mu^2 \frac{(\alpha + \rho)^2}{\alpha\rho(\alpha + \rho + 1)}$$

Value

dgwar gives the density, pgwar gives the distribution function, qgwar gives the quantile function, and rgwar generates random deviates.

The length of the result is determined by n for rgwar, and is the maximum of the lengths of the numerical arguments for the other functions.

Examples

```
dgwar(0, mu=1, k=2, rho=3)
pgwar(c(0,1,2,3), mu=1, k=2, rho=3)
qgwar(0.8, mu=1, k=2, rho=3)
rgwar(10, mu=1, k=2, rho=3)
```

`halton_dists` *Generate pseudo-random draws from specified distributions using Halton draws*

Description

Generate pseudo-random draws from specified distributions using Halton draws

Usage

```
halton_dists(dist, mean, sdev, hdraw = NULL, ndraws = 500)
```

Arguments

`dist`	The distribution type to use. The distribution options include normal ("n"), log-normal ("ln"), triangular ("t"), uniform ("u"), and gamma ("g").
`mean`	The mean value for the random draws.
`sdev`	The standard deviation value for the random draws.
`hdraw`	An optional vector of Halton draws to convert to the specified distribution. If not provided, the function will generate Halton draws.
`ndraws`	The number of random draws to generate. This is only used if 'hdraw' is not provided.

Details

This function is used to convert Halton draws to the specified distribution. The function can be used to generate random draws for use in random parameter models, generating Halton-based pseudo-random draws for specified distributions, etc.

The distributions generated all use the 'mean' (

$$\mu$$

) and 'sdev' (

$$\sigma$$

) parameters to generate the random draws. The density functions for the distributions are as follows: The Normal distribution is: $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

The Lognormal distribution is: $f(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log(x)-\mu)^2}{2\sigma^2}\right)$

The Triangular distribution is (note that this is a symmetrical triangular distribution where

$$\mu$$

is the median and

$$\sigma$$

is the half-width): $f(x) = \begin{cases} \frac{(x-\mu+\sigma)}{\sigma^2}, & \text{for } \mu - \sigma \leq x \leq \mu \\ \frac{(\mu+\sigma-x)}{\sigma^2}, & \text{for } \mu < x \leq \mu + \sigma \\ 0, & \text{otherwise} \end{cases}$

The Uniform distribution is (note that μ is the midpoint and σ is the half-width): $f(x) = \frac{1}{(\beta_\mu+\beta_\sigma)-(\beta_\mu-\beta_\sigma)} = \frac{1}{2\beta_\sigma}$

The Gamma distribution is based on

$$\mu = \frac{\alpha}{\beta}$$

and

$$\sigma^2 = \frac{\alpha}{\beta^2}$$

: $f(x) = \frac{\left(\frac{\mu}{\sigma^2}\right)^{\frac{\mu^2}{\sigma^2}}}{\Gamma\left(\frac{\mu^2}{\sigma^2}\right)} x^{\frac{\mu^2}{\sigma^2}-1} e^{-\frac{\mu}{\sigma^2}x}$

Value

A vector of psudo-random draws from the specified distribution, based on Halton draws.

Examples

```
# Generate 500 random draws from a normal distribution
halton_dists(dist="n", mean=3, sdev=2, ndraws=500)

# Generate 500 random draws from a lognormal distribution
halton_dists(dist="ln", mean=2, sdev=1.5, ndraws=500)

# Generate 500 random draws from a triangular distribution
halton_dists(dist="t", mean=1, sdev=0.5, ndraws=500)

# Generate 500 random draws from a uniform distribution
halton_dists(dist="u", mean=8, sdev=3, ndraws=500)

# Generate 500 random draws from a gamma distribution
halton_dists(dist="g", mean=0.5, sdev=1.5, ndraws=500)
```

---

invgamma *Inverse Gamma Distribution*

Description

These functions provide the density function, distribution function, quantile function, and random number generation for the Inverse-Gamma (IG) Distribution

Usage

```
dinvgamma(x, shape = 2.5, scale = 1, log = FALSE)

pinvgamma(q, shape = 2.5, scale = 1, lower.tail = TRUE, log.p = FALSE)

qinvgamma(p, shape = 2.5, scale = 1, lower.tail = TRUE, log.p = FALSE)

rinvgamma(n, shape = 2.5, scale = 1)
```

Arguments

x	numeric value or a vector of values.
shape	numeric value or vector of shape values for the distribution (the values have to be greater than 0).
scale	single value or vector of values for the scale parameter of the distribution (the values have to be greater than 0).
log	logical; if TRUE, probabilities p are given as log(p).
q	quantile or a vector of quantiles.
lower.tail	logical; if TRUE, probabilities p are $P[X \leq x]$ otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).
p	probability or a vector of probabilities.
n	the number of random numbers to generate.

Details

dinvgamma computes the density (PDF) of the Inverse-Gamma Distribution.

pinvgamma computes the CDF of the Inverse-Gamma Distribution.

qinvgamma computes the quantile function of the Inverse-Gamma Distribution.

rinvgamma generates random numbers from the Inverse-Gamma Distribution.

The compound Probability Mass Function (PMF) for the Inverse-Gamma distribution:

$$f(x|\alpha,\beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)}\left(\frac{1}{x}\right)^{\alpha+1} e^{-\frac{\beta}{x}}$$

Where α is the shape parameter and β is a scale parameter with the restrictions that $\alpha > 0$ and $\eta > 0$, and $x > 0$.

The CDF of the Inverse-Gamma distribution is:

$$F(x|\alpha,\beta) = \frac{\alpha.\Gamma\left(\frac{\beta}{x}\right)}{\Gamma(\alpha)} = Q\left(\alpha, \frac{\beta}{x}\right)$$

Where the numerator is the incomplete gamma function and $Q(\cdot)$ is the regularized gamma function.

The mean of the distribution is (provided $\alpha > 1$):

$$\mu = \frac{\beta}{\alpha - 1}$$

The variance of the distribution is (for $\alpha > 2$):

$$\sigma^2 = \frac{\beta^2}{(\alpha - 1)^2(\alpha - 2)}$$

Value

dinvgamma gives the density, pinvgamma gives the distribution function, qinvgamma gives the quantile function, and rinvgamma generates random deviates.

The length of the result is determined by n for rinvgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

Examples

```
dinvgamma(1, shape = 3, scale = 2)
pinvgamma(c(0.1, 0.5, 1, 3, 5, 10, 30), shape = 3, scale = 2)
qinvgamma(c(0.1, 0.3, 0.5, 0.9, 0.95), shape = 3, scale = 2)
rinvgamma(30, shape = 3, scale = 2)
```

---

mae *Calculate Mean Absolute Error (MAE)*

Description

This function calculates the Mean Absolute Error (MAE) between observed and predicted values.

Usage

```
mae(y, mu)
```

Arguments

y	Numeric vector representing the observed values.
mu	Numeric vector representing the predicted values.

Details

The MAE is calculated using the formula:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \mu_i|$$

Where y is the vector of observed values and μ is the vector of predicted values.

Value

Numeric value representing the MAE.

Examples

```
y <- c(1, 2, 3)
mu <- c(1.1, 1.9, 3.2)
mae(y, mu)
```

`mgf_lognormal` *Moment Generating Function for a Lognormal Distribution*

Description

Computes the value of the moment generating function (MGF) for a lognormal distribution at a given point through numerical integration. This function is particularly useful for distributions where the MGF does not have a closed-form solution. The lognormal distribution is specified by its log-mean (μ) and log-standard deviation (σ).

Usage

```
mgf_lognormal(mu, sigma, n)
```

Arguments

`mu`	The mean of the log-transformed variable, corresponding to μ in the lognormal distribution's parameters.
`sigma`	The standard deviation of the log-transformed variable, corresponding to σ in the lognormal distribution's parameters.
`n`	The point at which to evaluate the MGF, often denoted as t in the definition of the MGF. This parameter essentially specifies the order of the moment generating function.

Details

The moment generating function (MGF) for the lognormal distribution does not have a closed form solution. The MGF is defined as:

$$M_x(n) = \int_0^\infty e^{nx} \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}\, dx$$

The MGF for the lognormal distribution is useful for adjusting the predictions of generalized linear mixed models (GLMMs) that have parameters that follow a lognormal distribution and use a log link function. The adjustment for the mean value is the MGF with $n = 1$ or $E[e^x] = M_x(n = 1)$. The variance for the lognormal random parameter is:

$$Var(e^x) = E[e^{2x}] - E[e^x]^2 = M_x(n = 2) - M_x(n = 1)^2$$

Value

The estimated value of the moment generating function (MGF) for the specified lognormal distribution at the given point.

Examples

```
mu <- 0
sigma <- 1
n <- 1
mgf_value <- mgf_lognormal(mu, sigma, n)
print(mgf_value)
```

---

myAIC *Calculate Akaike Information Criterion (AIC)*

Description

This function calculates the Akaike Information Criterion (AIC) for a given model.

Usage

```
myAIC(LL, nparam)
```

Arguments

LL	Numeric value representing the log-likelihood of the model.
nparam	Numeric value representing the number of parameters in the model.

Details

The AIC is calculated using the formula:

$$AIC = -2 \cdot LL + 2 \cdot nparam$$

Where LL is the log-likelihood of the model and $nparam$ is the number of parameters.

Value

Numeric value representing the AIC.

Examples

```
LL <- -120.5
nparam <- 5
myAIC(LL, nparam)
```

myBIC *Calculate Bayesian Information Criterion (BIC)*

Description

This function calculates the Bayesian Information Criterion (BIC) for a given model.

Usage

```
myBIC(LL, nparam, n)
```

Arguments

LL	Numeric value representing the log-likelihood of the model.
nparam	Numeric value representing the number of parameters in the model.
n	Numeric value representing the number of observations.

Details

The BIC is calculated using the formula:

$$BIC = -2 \cdot LL + nparam \cdot \log(n)$$

Where LL is the log-likelihood of the model, $nparam$ is the number of parameters, and n is the number of observations.

Value

Numeric value representing the BIC.

Examples

```
LL <- -120.5
nparam <- 5
n <- 100
myBIC(LL, nparam, n)
```

NegativeBinomialLindley
Poisson-Lindley-Gamma (Negative Binomial-Lindley) Distribution

Description

These functions provide density, distribution function, quantile function, and random number generation for the Poisson-Lindley-Gamma (PLG) Distribution

Usage

```
dplindGamma(x, mean = 1, theta = 1, alpha = 1, log = FALSE)

pplindGamma(
  q,
  mean = 1,
  theta = 1,
  alpha = 1,
  lower.tail = TRUE,
  log.p = FALSE
)

qplindGamma(p, mean = 1, theta = 1, alpha = 1)

rplindGamma(n, mean = 1, theta = 1, alpha = 1)
```

Arguments

x	numeric value or a vector of values.
mean	numeric value or vector of mean values for the distribution (the values have to be greater than 0).
theta	single value or vector of values for the theta parameter of the distribution (the values have to be greater than 0).
alpha	single value or vector of values for the 'alpha' parameter of the gamma distribution in the special case that the mean = 1 and the variance = 'alpha' (the values for 'alpha' have to be greater than 0).
log	logical; if TRUE, probabilities p are given as log(p).
q	quantile or a vector of quantiles.
lower.tail	logical; if TRUE, probabilities p are $P[X \leq x]$ otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).
p	probability or a vector of probabilities.
n	the number of random numbers to generate.

Details

The Poisson-Lindley-Gamma is a count distribution that captures high densities for small integer values and provides flexibility for heavier tails.

`dplindGamma` computes the density (PDF) of the Poisson-Lindley-Gamma Distribution.

`pplindGamma` computes the CDF of the Poisson-Lindley-Gamma Distribution.

`qplindGamma` computes the quantile function of the Poisson-Lindley-Gamma Distribution.

`rplindGamma` generates random numbers from the Poisson-Lindley-Gamma Distribution.

The compound Probability Mass Function (PMF) for the Poisson-Lindley-Gamma (PLG) distribution is:

$$f(x|\mu,\theta,\alpha) = \frac{\alpha(\theta+2)^2\Gamma(x+\alpha)}{\mu^2(\theta+1)^3\Gamma(\alpha)}\left(\frac{\mu\theta(\theta+1)}{\theta+2}U\left(x+1,2-\alpha,\frac{\alpha(\theta+2)}{\mu(\theta+1)}\right)+\alpha(x+1)U\left(x+2,3-\alpha,\frac{\alpha(\theta+2)}{\mu(\theta+1)}\right)\right)$$

Where θ is a distribution parameter from the Poisson-Lindley distribution with the restrictions that $\theta > 0$, α is a parameter for the gamma distribution with the restriction $\alpha > 0$, mu is the mean value, and x is a non-negative integer, and

$$U(a,b,z)$$

is the Tricomi's solution to the confluent hypergeometric function - also known as the confluent hypergeometric function of the second kind

The expected value of the distribution is:

$$E[x] = \mu$$

The variance is:

$$\sigma^2 = \mu + \left(2\alpha + 1 - \frac{2(1+\alpha)}{(\theta+2)^2}\right)\mu^2$$

While the distribution can be computed using the confluent hypergeometric function, that function has limitations in value it can be computed at (along with accuracy, in come cases). For this reason, the function uses Halton draws to perform simulation over the gamma distribution to solve the integral. This is sometimes more computationally efficient as well.

Value

dplindGamma gives the density, pplindGamma gives the distribution function, qplindGamma gives the quantile function, and rplindGamma generates random deviates.

The length of the result is determined by n for rplindGamma, and is the maximum of the lengths of the numerical arguments for the other functions.

Examples

```
dplindGamma(0, mean=0.75, theta=7, alpha=2)
pplindGamma(c(0,1,2,3,5,7,9,10), mean=0.75, theta=3, alpha=0.5)
qplindGamma(c(0.1,0.3,0.5,0.9,0.95), mean=1.67, theta=0.5, alpha=0.5)
rplindGamma(30, mean=0.5, theta=0.5, alpha=2)
```

poisLindRE *Function for estimating a Random Effects Poisson-Lindley regression model*

Description

Function for estimating a Random Effects Poisson-Lindley regression model

Usage

```
poisLind.re(
  formula,
  group_var,
  data,
  method = "NM",
  max.iters = 1000,
  print.level = 0,
  bootstraps = NULL,
  offset = NULL
)
```

Arguments

formula	an R formula.
group_var	the grouping variable(s) indicating random effects (e.g., individual ID).
data	a dataframe that has all of the variables in the formula.
method	a method to use for optimization in the maximum likelihood estimation. For options, see maxLik. Note that "BHHH" is not available for this function due to the implementation for the random effects.
max.iters	the maximum number of iterations to allow the optimization method to perform.
print.level	Integer specifying the verbosity of output during optimization.
bootstraps	Optional integer specifying the number of bootstrap samples to be used for estimating standard errors. If not specified, no bootstrapping is performed.
offset	an optional offset term provided as a string.

Details

The function poisLindRE is similar to the poisLind function, but it includes an additional argument group_var that specifies the grouping variable for the random effects. The function estimates a Random Effects Poisson-Lindley regression model using maximum likelihood. It is similar to poisLind, but includes additional terms to account for the random effects.

The Random Effects Poisson-Lindley model is useful for panel data and assumes that the random effects follow a gamma distribution. The PDF is

$$f(y_{it}|\mu_{it},\theta) = \frac{\theta^2}{\theta+1}\prod_{t=1}^{n_i}\frac{\left(\mu_{it}\frac{\theta(\theta+1)}{\theta+2}\right)^{y_{it}}}{y_{it}!}\cdot\frac{(\sum_{t=1}^{n_i}y_{it})!\left(\sum_{t=1}^{n_i}\mu_{it}\frac{\theta(\theta+1)}{\theta+2}+\theta+\sum_{t=1}^{n_i}y_{it}+1\right)}{\left(\sum_{t=1}^{n_i}\mu_{it}\frac{\theta(\theta+1)}{\theta+2}+\theta\right)^{\sum_{t=1}^{n_i}y_{it}+2}}$$

The log-likelihood function is:

$$LL = 2\log(\theta) - \log(\theta+1) + \sum_{t=1}^{n_i} y_{it}\log(\mu_{it}) + \sum_{t=1}^{n_i} y_{it}\log\left(\frac{\theta(\theta+1)}{\theta+2}\right) - \sum_{t=1}^{n_i}\log(y_{it}!) + \log\left(\left(\sum_{t=1}^{n_i} y_{it}\right)!\right) + \log\left(\sum_{t=1}^{n_i} \mu_{it}\frac{\theta(\theta -$$

The mean and variance are:

$$\mu_{it} = \exp(X_{it}\beta)$$

$$V(\mu_{it}) = \mu_{it} + \left(1 - \frac{2}{(\theta+2)^2}\right)\mu_{it}^2$$

Value

An object of class 'countreg' which is a list with the following components:

- model: the fitted model object.
- data: the data frame used to fit the model.
- call: the matched call.
- formula: the formula used to fit the model.

Examples

```
data("washington_roads")
washington_roads$AADTover10k <-
  ifelse(washington_roads$AADT > 10000, 1, 0)

poislind.mod <- poisLind.re(
  Animal ~ lnaadt + lnlength + speed50 +
    ShouldWidth04 + AADTover10k,
  data      = washington_roads,
  group_var = "ID",
  method    = "NM",
  max.iters = 1000
)
summary(poislind.mod)
```

PoissonGeneralizedExponential

Poisson-Generalized-Exponential Distribution

Description

These functions provide density, distribution function, quantile function, and random number generation for the Poisson-Generalized-Exponential (PGE) Distribution

Usage

```
dpge(
  x,
  mean = 1,
  shape = 1,
  scale = 1,
  ndraws = 1500,
  log = FALSE,
  haltons = NULL
)

ppge(
  q,
  mean = 1,
  shape = 1,
  scale = 1,
  ndraws = 1500,
  lower.tail = TRUE,
  log.p = FALSE,
  haltons = NULL
)

qpge(p, mean = 1, shape = 1, scale = 1, ndraws = 1500)

rpge(n, mean = 1, shape = 1, scale = 1, ndraws = 1500)
```

Arguments

x	numeric value or a vector of values.
mean	numeric value or vector of mean values for the distribution (the values have to be greater than 0). This is NOT the value of λ.
shape	numeric value or vector of shape values for the shape parameter of the generalized exponential distribution (the values have to be greater than 0).
scale	single value or vector of values for the scale parameter of the generalized exponential distribution (the values have to be greater than 0).
ndraws	the number of Halton draws to use for the integration.
log	logical; if TRUE, probabilities p are given as log(p).
haltons	an optional vector of Halton draws to use instead of ndraws.
q	quantile or a vector of quantiles.
lower.tail	logical; if TRUE, probabilities p are $P[X \leq x]$ otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).
p	probability or a vector of probabilities.
n	the number of random numbers to generate.

Details

`dpge` computes the density (PDF) of the PGE Distribution.

`ppge` computes the CDF of the PGE Distribution.

`qpge` computes the quantile function of the PGE Distribution.

`rpge` generates random numbers from the PGE Distribution.

The Generalized Exponential distribution can be written as a function with a shape parameter $\alpha > 0$ and scale parameter $\gamma > 0$. The distribution has strictly positive continuous values. The PDF of the distribution is:

$$f(x|\alpha,\gamma) = \frac{\alpha}{\gamma}\left(1 - e^{-\frac{x}{\gamma}}\right)^{\alpha-1} e^{-\frac{x}{\gamma}}$$

Thus, the compound Probability Mass Function(PMF) for the PGE distribution is:

$$f(y|\lambda,\alpha,\beta) = \int_0^\infty \frac{\lambda^y x^y e^{-\lambda x}}{y!}\frac{\alpha}{\gamma}\left(1 - e^{-\frac{x}{\gamma}}\right)^{\alpha-1} e^{-\frac{x}{\gamma}}\,dx$$

The expected value of the distribution is:

$$E[y] = \mu = \lambda\left(\frac{\psi(\alpha+1) - \psi(1)}{\gamma}\right)$$

Where $\psi(\cdot)$ is the digamma function.

The variance is:

$$\sigma^2 = \lambda\left(\frac{\psi(\alpha+1) - \psi(1)}{\gamma}\right) + \left(\frac{-\psi'(\alpha+1) + \psi'(1)}{\gamma^2}\right)\lambda^2$$

Where $\psi'(\cdot)$ is the trigamma function.

To ensure that $\mu = e^{X\beta}$, λ is replaced with:

$$\lambda = \frac{\gamma e^{X\beta}}{\psi(\alpha+1) - \psi(1)}$$

This results in:

$$f(y|\mu,\alpha,\beta) = \int_0^\infty \frac{\left(\frac{\gamma e^{X\beta}}{\psi(\alpha+1)-\psi(1)}\right)^y x^y e^{-\left(\frac{\gamma e^{X\beta}}{\psi(\alpha+1)-\psi(1)}\right)x}}{y!}\frac{\alpha}{\gamma}\left(1 - e^{-\frac{x}{\gamma}}\right)^{\alpha-1} e^{-\frac{x}{\gamma}}\,dx$$

Halton draws are used to perform simulation over the lognormal distribution to solve the integral.

Value

dpge gives the density, ppge gives the distribution function, qpge gives the quantile function, and rpge generates random deviates.

The length of the result is determined by n for rpge, and is the maximum of the lengths of the numerical arguments for the other functions.

References

Gupta, R. D., & Kundu, D. (2007). Generalized exponential distribution: Existing results and some recent developments. Journal of Statistical planning and inference, 137(11), 3537-3547.

Examples

```
dpge(0, mean=0.75, shape=2, scale=1, ndraws=2000)
ppge(c(0,1,2,3,4,5,6), mean=0.75, shape=2, scale=1, ndraws=500)
qpge(c(0.1,0.3,0.5,0.9,0.95), mean=0.75, shape=2, scale=1, ndraws=500)
rpge(30, mean=0.75,  shape=2, scale=1, ndraws=500)
```

PoissonInverseGamma *Poisson-Inverse-Gamma Distribution*

Description

These functions provide the density function, distribution function, quantile function, and random number generation for the Poisson-Inverse-Gamma (PInvGamma) Distribution

Usage

```
dpinvgamma(x, mu = 1, eta = 1, log = FALSE)

ppinvgamma(q, mu = 1, eta = 1, lower.tail = TRUE, log.p = FALSE)

qpinvgamma(p, mu = 1, eta = 1)

rpinvgamma(n, mu = 1, eta = 1)
```

Arguments

x	numeric value or a vector of values.
mu	numeric value or vector of mean values for the distribution (the values have to be greater than 0).
eta	single value or vector of values for the scale parameter of the distribution (the values have to be greater than 0).
log	logical; if TRUE, probabilities p are given as log(p).
q	quantile or a vector of quantiles.
lower.tail	logical; if TRUE, probabilities p are $P[X \leq x]$ otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).
p	probability or a vector of probabilities.
n	the number of random numbers to generate.

Details

dpinvgamma computes the density (PDF) of the Poisson-Inverse-Gamma Distribution.

ppinvgamma computes the CDF of the Poisson-Inverse-Gama Distribution.

qpinvgamma computes the quantile function of the Poisson-Inverse-Gamma Distribution.

rpinvgamma generates random numbers from the Poisson-Inverse-Gamma Distribution.

The compound Probability Mass Function (PMF) for the Poisson-Inverse-Gamma distribution is:

$$f(x|\eta,\mu) = \frac{2\left(\mu\left(\frac{1}{\eta}+1\right)\right)^{\frac{x+\frac{1}{eta}+2}{2}}}{x!\Gamma\left(\frac{1}{\eta}+2\right)} K_{x-\frac{1}{\eta}-2}\left(2\sqrt{\mu\left(\frac{1}{\eta}+1\right)}\right)$$

Where η is a shape parameter with the restriction that $\eta > 0$, $\mu > 0$ is the mean value, y is a non-negative integer, and $K_i(z)$ is the modified Bessel function of the second kind. This formulation uses the mean directly.

The variance of the distribution is:

$$\sigma^2 = \mu + \eta\mu^2$$

Value

dpinvgamma gives the density, ppinvgamma gives the distribution function, qpinvgamma gives the quantile function, and rcom generates random deviates.

The length of the result is determined by n for rpinvgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

Examples

```
dpinvgamma(1, mu=0.75, eta=1)
ppinvgamma(c(0,1,2,3,5,7,9,10), mu=0.75, eta=3)
qpinvgamma(c(0.1,0.3,0.5,0.9,0.95), mu=0.75, eta=0.5)
rpinvgamma(30, mu=0.75, eta=1.5)
```

PoissonInverseGaussian

Poisson-Inverse-Gaussian Distribution

Description

These functions provide the density function, distribution function, quantile function, and random number generation for the Poisson-Inverse-Gaussian (PInvGaus) Distribution.

These functions provide the density function, distribution function, quantile function, and random number generation for the Poisson-Inverse-Gaussian (PInvGaus) Distribution

Usage

```
dpinvgaus(x, mu = 1, eta = 1, form = "Type 1", log = FALSE)

ppinvgaus(
  q,
  mu = 1,
  eta = 1,
  form = "Type 1",
  lower.tail = TRUE,
  log.p = FALSE
)

qpinvgaus(p, mu = 1, eta = 1, form = "Type 1")

rpinvgaus(n, mu = 1, eta = 1, form = "Type 1")

dpinvgaus(x, mu = 1, eta = 1, form = "Type 1", log = FALSE)

ppinvgaus(
  q,
  mu = 1,
  eta = 1,
  form = "Type 1",
  lower.tail = TRUE,
  log.p = FALSE
)

qpinvgaus(p, mu = 1, eta = 1, form = "Type 1")

rpinvgaus(n, mu = 1, eta = 1, form = "Type 1")
```

Arguments

x	numeric value or a vector of values.
mu	numeric value or vector of mean values for the distribution (the values have to be greater than 0).
eta	single value or vector of values for the scale parameter of the distribution (the values have to be greater than 0).
form	optional parameter indicating which formulation to use. Options include "Type 1" which is the standard form or "Type 2" which follows the formulation by Dean et. al. (1987).
log	logical; if TRUE, probabilities p are given as log(p).
q	quantile or a vector of quantiles.
lower.tail	logical; if TRUE, probabilities p are $P[X \leq x]$ otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).
p	probability or a vector of probabilities.
n	the number of random numbers to generate.

Details

The Poisson-Inverse-Gaussian distribution is a special case of the Sichel distribution (Cameron & Trivedi, 2013). It is also known as a univariate Sichel distribution (Hilbe, 2011).

`dpinvgaus` computes the PDF of the Poisson-Inverse-Gaussian dist.

`ppinvgaus` computes the CDF of the Poisson-Inverse-Gaussian dist.

`qpinvgaus` computes quantiles of the Poisson-Inverse-Gaussian dist.

`rpinvgaus` generates random numbers from the distribution.

The PMF (Type 1) is:

$$f(y|\eta,\mu) = \begin{cases} f(0) = \exp\left(\frac{\mu}{\eta}(1-\sqrt{1+2\eta})\right) \\ f(y>0) = f(0)\frac{\mu^y}{y!}(1+2\eta)^{-y/2}\sum_{j=0}^{y-1}\frac{\Gamma(y+j)}{\Gamma(y-j)\Gamma(j+1)}\left(\frac{\eta}{2\mu}\right)^j(1+2\eta)^{-j/2} \end{cases}$$

The variance is:

$$\sigma^2 = \mu + \eta\mu$$

Type 2 modifies $\eta \to \eta\mu$:

$$f(0) = \exp\left(\frac{1}{\eta}(1-\sqrt{1+2\eta\mu})\right)$$

$$f(y>0) = f(0)\frac{\mu^y}{y!}(1+2\eta\mu)^{-y/2}\sum_{j=0}^{y-1}\frac{\Gamma(y+j)}{\Gamma(y-j)\Gamma(j+1)}\left(\frac{\eta}{2}\right)^j(1+2\eta\mu)^{-j/2}$$

Resulting variance:

$$\sigma^2 = \mu + \eta\mu^2$$

The Poisson-Inverse-Gaussian distribution is a special case of the Sichel distribution, as noted by Cameron & Trivedi (2013). It is also known as a univariate Sichel distribution (Hilbe, 2011).

`dpinvgaus` computes the density (PDF) of the Poisson-Inverse-Gaussian Distribution.

`ppinvgaus` computes the CDF of the Poisson-Inverse-Gaussian Distribution.

`qpinvgaus` computes the quantile function of the Poisson-Inverse-Gaussian Distribution.

`rpinvgaus` generates random numbers from the Poisson-Inverse-Gamma Distribution.

The compound Probability Mass Function (PMF) for the Poisson-Inverse-Gaussian distribution (Type 1) is (Cameron & Trivedi, 2013):

$$f(y|\eta,\mu) = \begin{cases} f(y=0) = \exp\left(\frac{\mu}{\eta}\left(1-\sqrt{1+2\eta}\right)\right) \\ f(y|y>0) = f(y=0)\frac{\mu^y}{y!}(1+2\eta)^{-y/2}\cdot\sum_{j=0}^{y-1}\frac{\Gamma(y+j)}{\Gamma(y-j)\Gamma(j+1)}\left(\frac{\eta}{2\mu}\right)^2(1+2\eta)^{-j/2} \end{cases}$$

Where η is a scale parameter with the restriction that $\eta > 0$, μ is the mean value, and y is a non-negative integer.

The variance of the distribution is:

$$\sigma^2 = \mu + \eta\mu$$

The alternative parametrization by Dean et. al. (1987) replaces η with $\eta\mu$. This version (Type 2) has the PMF:

$$f(y|\eta,\mu) = \begin{cases} f(y=0) = \exp\left(\frac{1}{\eta}\left(1-\sqrt{1+2\eta\mu}\right)\right) \\ f(y|y>0) = f(y=0)\frac{\mu^y}{y!}(1+2\eta\mu)^{-y/2}\cdot\sum_{j=0}^{y-1}\frac{\Gamma(y+j)}{\Gamma(y-j)\Gamma(j+1)}\left(\frac{\eta}{2}\right)^2(1+2\eta\mu)^{-j/2} \end{cases}$$

This results in the variance of:

$$\sigma^2 = \mu + \eta\mu^2$$

Value

description dpinvgaus gives the density, ppinvgaus gives the distribution function, qpinvgaus gives the quantile function, and rpinvgaus generates random deviates.

The length of the result is determined by n for rpinvgaus, and is the maximum of the lengths of the numerical arguments for the other functions.

dpinvgaus gives the density, ppinvgaus gives the distribution function, qpinvgaus gives the quantile function, and rpinvgaus generates random deviates.

The length of the result is determined by n for rpinvgaus, and is the maximum of the lengths of the numerical arguments for the other functions.

References

Cameron & Trivedi (2013). Regression Analysis of Count Data. Dean, Lawless & Willmot (1989). Mixed Poisson–Inverse Gaussian Models. Hilbe (2011). Negative Binomial Regression.

Cameron, A. C., & Trivedi, P. K. (2013). Regression analysis of count data, 2nd Edition. Cambridge university press.

Dean, C., Lawless, J. F., & Willmot, G. E. (1989). A mixed Poisson–Inverse-Gaussian regression model. Canadian Journal of Statistics, 17(2), 171-181.

Hilbe, J. M. (2011). Negative binomial regression. Cambridge University Press.

Examples

```
dpinvgaus(1, mu=0.75, eta=1)
ppinvgaus(c(0,1,2,3,5,7,9,10), mu=0.75, eta=3, form="Type 2")
qpinvgaus(c(0.1,0.3,0.5,0.9,0.95), mu=0.75, eta=0.5, form="Type 2")
rpinvgaus(30, mu=0.75, eta=1.5)
```

```
dpinvgaus(1, mu=0.75, eta=1)
ppinvgaus(c(0,1,2,3,5,7,9,10), mu=0.75, eta=3, form="Type 2")
qpinvgaus(c(0.1,0.3,0.5,0.9,0.95), mu=0.75, eta=0.5, form="Type 2")
rpinvgaus(30, mu=0.75, eta=1.5)
```

`PoissonLindley` *Poisson-Lindley Distribution*

Description

These functions provide density, distribution function, quantile function, and random number generation for the Poisson-Lindley (PL) Distribution

Usage

```
msg1

dplind(x, mean = 1, theta = 1, lambda = NULL, log = FALSE)

pplind(q, mean = 1, theta = 1, lambda = NULL, lower.tail = TRUE, log.p = FALSE)

qplind(p, mean = 1, theta = 1, lambda = NULL)

rplind(n, mean = 1, theta = 1, lambda = NULL)
```

Arguments

`x`	numeric value or a vector of values.
`mean`	numeric value or vector of mean values for the distribution (the values have to be greater than 0).
`theta`	single value or vector of values for the theta parameter of the distribution (the values have to be greater than 0).
`lambda`	alternative parameterization (use instead of the mean); numeric value or vector of values for lambda parameter of the distribution (the values have to be greater than 0).
`log`	logical; if TRUE, probabilities p are given as log(p).
`q`	quantile or a vector of quantiles.
`lower.tail`	logical; if TRUE, probabilities p are $P[X \leq x]$ otherwise, $P[X > x]$.
`log.p`	logical; if TRUE, probabilities p are given as log(p).
`p`	probability or a vector of probabilities.
`n`	the number of random numbers to generate.

Format

An object of class `character` of length 1.

Details

The Poisson-Lindley is a 2-parameter count distribution that captures high densities for small integer values. This makes it ideal for data that are zero-inflated.

`dplind` computes the density (PDF) of the Poisson-Lindley Distribution.

`pplind` computes the CDF of the Poisson-Lindley Distribution.

`qplind` computes the quantile function of the Poisson-Lindley Distribution.

`rplind` generates random numbers from the Poisson-Lindley Distribution.

The compound Probability Mass Function (PMF) for the Poisson-Lindley (PL) distribution is:

$$f(y|\theta,\lambda) = \frac{\theta^2\lambda^y(\theta+\lambda+y+1)}{(\theta+1)(\theta+\lambda)^{y+2}}$$

Where θ and λ are distribution parameters with the restrictions that $\theta > 0$ and $\lambda > 0$, and y is a non-negative integer.

The expected value of the distribution is:

$$\mu = \frac{\lambda(\theta+2)}{\theta(\theta+1)}$$

The default is to use the input mean value for the distribution. However, the lambda parameter can be used as an alternative to the mean value.

Value

dplind gives the density, pplind gives the distribution function, qplind gives the quantile function, and rplind generates random deviates.

The length of the result is determined by n for rplind, and is the maximum of the lengths of the numerical arguments for the other functions.

Examples

```
dplind(0, mean = 0.75, theta = 7)
pplind(c(0, 1, 2, 3, 5, 7, 9, 10), mean = 0.75, theta = 7)
qplind(c(0.1, 0.3, 0.5, 0.9, 0.95), lambda = 4.67, theta = 7)
rplind(30, mean = 0.75, theta = 7)
```

PoissonLindleyLognormal

Poisson-Lindley-Lognormal Distribution

Description

These functions provide density, distribution, quantile, and random generation for the Poisson-Lindley-Lognormal (PLL) Distribution.

Usage

```
dplindLnorm(
  x,
  mean = 1,
  theta = 1,
  sigma = 1,
  ndraws = 1500,
  log = FALSE,
  hdraws = NULL
)

pplindLnorm(
  q,
  mean = 1,
  theta = 1,
  lambda = NULL,
  sigma = 1,
  ndraws = 1500,
  lower.tail = TRUE,
  log.p = FALSE
)

qplindLnorm(p, mean = 1, theta = 1, sigma = 1, ndraws = 1500, lambda = NULL)

rplindLnorm(n, mean = 1, theta = 1, sigma = 1, ndraws = 1500, lambda = NULL)
```

Arguments

x	numeric value or vector of values.
mean	mean (>0).
theta	Poisson-Lindley theta parameter (>0).
sigma	lognormal sigma parameter (>0).
ndraws	number of Halton draws.
log	return log-density.
hdraws	optional Halton draws.
q	quantile or vector of quantiles.
lambda	optional lambda parameter (>0).
lower.tail	TRUE returns P[X $\leq$ x].
log.p	return log-CDF.
p	probability or vector of probabilities.
n	number of random draws.

Details

The PLL is a 3-parameter count distribution that captures high mass at small y and allows flexible heavy tails.

`dplind` computes the PLL density. `pplind` computes the PLL CDF. `qplind` computes quantiles. `rplind` generates random draws.

The PMF is:

$$f(y|\mu,\theta,\sigma) = \int_0^\infty \frac{\theta^2 \mu^y x^y (\theta + \mu x + y + 1)}{(\theta+1)(\theta+\mu x)^{y+2}} \frac{\exp\left(-\frac{\ln^2(x)}{2\sigma^2}\right)}{x\sigma\sqrt{2\pi}} dx$$

Mean:

$$E[y] = \mu = \frac{\lambda(\theta+2)e^{\sigma^2/2}}{\theta(\theta+1)}$$

Halton draws are used to evaluate the integral.

Value

dplindLnorm gives the density, pplindLnorm gives the distribution function, qplindLnorm gives the quantile function, and rplindLnorm generates random deviates.

The length of the result is determined by n for rplindLnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

Examples

```
dplindLnorm(0, mean=0.75, theta=7, sigma=2, ndraws=10)
pplindLnorm(0:10, mean=0.75, theta=7, sigma=2, ndraws=10)
qplindLnorm(c(0.1,0.5,0.9), lambda=4.67, theta=7, sigma=2)
rplindLnorm(5, mean=0.75, theta=7, sigma=2)
```

PoissonLognormal *Poisson-Lognormal Distribution*

Description

These functions provide density, distribution function, quantile function, and random number generation for the Poisson-Lognormal (PLogN) Distribution

Usage

```
dpLnorm(x, mean = 1, sigma = 1, ndraws = 1500, log = FALSE, hdraws = NULL)

ppLnorm(
  q,
  mean = 1,
```

```
  sigma = 1,
  ndraws = 1500,
  lower.tail = TRUE,
  log.p = FALSE
)

qpLnorm(p, mean = 1, sigma = 1, ndraws = 1500)

rpLnorm(n, mean = 1, sigma = 1, ndraws = 1500)
```

Arguments

x	numeric value or a vector of values.
mean	numeric value or vector of mean values for the distribution (the values have to be greater than 0).
sigma	single value or vector of values for the sigma parameter of the lognormal distribution (the values have to be greater than 0).
ndraws	the number of Halton draws to use for the integration.
log	logical; if TRUE, probabilities p are given as log(p).
hdraws	and optional vector of Halton draws to use for the integration.
q	quantile or a vector of quantiles.
lower.tail	logical; if TRUE, probabilities p are $P[X \leq x]$ otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).
p	probability or a vector of probabilities.
n	the number of random numbers to generate.

Details

dpLnorm computes the density (PDF) of the Poisson-Lognormal Distribution.

ppLnorm computes the CDF of the Poisson-Lognormal Distribution.

qpLnorm computes the quantile function of the Poisson-Lognormal Distribution.

rpLnorm generates random numbers from the Poisson-Lognormal Distribution.

The compound Probability Mass Function (PMF) for the Poisson-Lognormal distribution is:

$$f(y|\mu,\theta,\alpha) = \int_0^\infty \frac{\mu^y x^y e^{-\mu x}}{y!} \frac{exp\left(-\frac{ln^2(x)}{2\sigma^2}\right)}{x\sigma\sqrt{2\pi}} dx$$

Where σ is a parameter for the lognormal distribution with the restriction $\sigma > 0$, and y is a non-negative integer.

The expected value of the distribution is:

$$E[y] = e^{X\beta+\sigma^2/2} = \mu e^{\sigma^2/2}$$

Halton draws are used to perform simulation over the lognormal distribution to solve the integral.

Value

dpLnorm gives the density, ppLnorm gives the distribution function, qpLnorm gives the quantile function, and rpLnorm generates random deviates.

The length of the result is determined by n for rpLnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

Examples

```
dpLnorm(0, mean=0.75, sigma=2, ndraws=10)
ppLnorm(c(0,1,2,3,5,7,9,10), mean=0.75, sigma=2, ndraws=10)
qpLnorm(c(0.1,0.3,0.5,0.9,0.95), mean=0.75, sigma=2, ndraws=10)
rpLnorm(30, mean=0.75,  sigma=2, ndraws=10)
```

PoissonWeibull *Poisson-Weibull Distribution Functions*

Description

These functions provide density, distribution function, quantile function, and random number generation for the Poisson-Weibull Distribution, which is specified either by its shape and scale parameters or by its mean and standard deviation.

Usage

```
dpoisweibull(
  x,
  lambda = NULL,
  alpha = NULL,
  sigma = NULL,
  mean_value = NULL,
  sd_value = NULL,
  ndraws = 1500,
  log = FALSE
)

ppoisweibull(
  q,
  lambda = NULL,
  alpha = NULL,
  sigma = NULL,
  mean_value = NULL,
  sd_value = NULL,
  ndraws = 1500,
  lower.tail = TRUE,
  log.p = FALSE
)
```

```
qpoisweibull(
  p,
  lambda = NULL,
  alpha = NULL,
  sigma = NULL,
  mean_value = NULL,
  sd_value = NULL,
  ndraws = 1500
)

rpoisweibull(
  n,
  lambda = NULL,
  alpha = NULL,
  sigma = NULL,
  mean_value = NULL,
  sd_value = NULL,
  ndraws = 1500
)
```

Arguments

x	A numeric value or vector of values for which the PDF or CDF is calculated.
lambda	Mean value of the Poisson distribution.
alpha	Shape parameter of the Weibull distribution (optional if mean and sd are provided).
sigma	Scale parameter of the Weibull distribution (optional if mean and sd are provided).
mean_value	Mean of the Weibull distribution (optional if alpha and sigma are provided).
sd_value	Standard deviation of the Weibull distribution (optional if alpha and sigma are provided).
ndraws	the number of Halton draws to use for the integration.
log	Logical; if TRUE, probabilities p are given as log(p).
q	Quantile or a vector of quantiles.
lower.tail	Logical; if TRUE, probabilities are P[X <= x], otherwise P[X > x].
log.p	Logical; if TRUE, probabilities p are given as log(p).
p	A numeric value or vector of probabilities for the quantile function.
n	The number of random samples to generate.

Details

The Poisson-Weibull distribution uses the Weibull distribution as a mixing distribution for a Poisson process. It is useful for modeling overdispersed count data. The density function (probability mass

function) for the Poisson-Weibull distribution is given by:

$$P(y|\lambda,\alpha,\sigma) = \int_0^\infty \frac{e^{-\lambda x}\lambda^y x^y}{y!}\left(\frac{\alpha}{\sigma}\right)\left(\frac{x}{\sigma}\right)^{\alpha-1} e^{-\left(\frac{x}{\sigma}\right)^\alpha} dx$$

where $f(x|\alpha,\sigma)$ is the PDF of the Weibull distribution and λ is the mean of the Poisson distribution.

`dpoisweibull` computes the density of the Poisson-Weibull distribution.

`ppoisweibull` computes the distribution function of the Poisson-Weibull distribution.

`qpoisweibull` computes the quantile function of the Poisson-Weibull distribution.

`rpoisweibull` generates random numbers following the Poisson-Weibull distribution.

The shape and scale parameters directly define the Weibull distribution, whereas the mean and standard deviation are used to compute these parameters indirectly.

Value

dpoisweibull gives the density, ppoisweibull gives the distribution function, qpoisweibull gives the quantile function, and rpoisweibull generates random deviates.

The length of the result is determined by n for rpoisweibull, and is the maximum of the lengths of the numerical arguments for the other functions.

Examples

```
dpoisweibull(4, lambda=1.5, mean_value=1.5, sd_value=0.5, ndraws=10)
ppoisweibull(4, lambda=1.5, mean_value=1.5, sd_value=2, ndraws=10)
qpoisweibull(0.95, lambda=1.5, mean_value=1.5, sd_value=2, ndraws=10)
rpoisweibull(10, lambda=1.5, mean_value=1.5, sd_value=2, ndraws=10)
```

`predict.flexCountReg` *Predictions for flexCountReg models*

Description

Generates predictions for the expected count (lambda) for observations.

For **countreg.rp** (Random Parameters) models, three methods are available:

- **Simulated**: Uses Halton draws to simulate the random parameters and averages the outcomes. This is a simulation-based approximation.
- **Individual**: Estimates observation-specific coefficients (conditional on observed outcomes) using Empirical Bayes. Requires the outcome variable to be present in `data`.
- **Exact**: Uses the analytical Moment Generating Functions (MGFs) of the random parameter distributions to calculate the exact expected value. This method is faster and removes simulation error.

For **countreg**, **poisLindRE**, and **RENB** models, the function calculates the expected value $\mu = \exp(X\beta)$ (with appropriate adjustments for specific families like PLN or underreporting).

Usage

```
## S3 method for class 'flexCountReg'
predict(object, newdata = NULL, ...)
```

Arguments

object	a model object estimated using this R package.
newdata	optional dataframe for which to generate predictions.
...	optional arguments passed to the function. This includes 'method'.

Value

A numeric vector of predicted expected counts for each observation in the provided data. If no data is provided, the predictions for the data used in estimating the model are provided.

Note

optional parameter 'newdata': a dataframe that has all of the variables in the `formula` and `rpar_formula`.

optional parameter 'method': Only valid for random parameters models ('countreg.rp'). Options include `Simulated` (default), `Individual`, or `Exact`.

References

Wood, J.S., Gayah, V. (2025). Out-of-sample prediction and interpretation for random parameter generalized linear models. *Accident Analysis and Prevention*, 220, 108147.

Examples

```
# Load data and create a dummy variable
data("washington_roads")
washington_roads$AADT10kplus <- ifelse(washington_roads$AADT > 10000, 1, 0)

# ============================================================================
# 1. Fixed Parameter Model (countreg)
# ============================================================================
nb2_fixed <- countreg(Total_crashes ~ lnaadt + lnlength + speed50,
                      data = washington_roads,
                      family = "NB2")
pred_fixed <- predict(nb2_fixed, data = washington_roads)

# ============================================================================
# 2. Random Parameters Model (countreg.rp)
# ============================================================================
rp_nb2 <- countreg.rp(Total_crashes ~ lnaadt + lnlength,
                      rpar_formula = ~ -1 + speed50,
                      data = washington_roads,
                      family = "NB2",
                      rpardists = c(speed50 = "n"),
                      ndraws = 100)
```

```
# Method A: Simulated (Default)
pred_sim <- predict(rp_nb2, data = washington_roads, method = "Simulated")

# Method B: Exact (Analytical MGF)
pred_exact <- predict(rp_nb2, data = washington_roads, method = "Exact")

# ============================================================================
# 3. Random Effects Models (poisLindRE / RENB)
# ============================================================================
pl_re <- poisLind.re(Total_crashes ~ lnaadt + lnlength,
                     data = washington_roads,
                     group_var = "ID")
pred_pl_re <- predict(pl_re, data = washington_roads)
```

`regCompTable` *Create a Table Comparing Regression Models with AIC, BIC, and McFadden's Pseudo-R-Squared*

Description

This function creates tables comparing the flexCountReg package models supplied to the function.

Usage

```
regCompTable(
  models,
  coefs = TRUE,
  AIC = TRUE,
  BIC = TRUE,
  RSquare = TRUE,
  tableType = "tibble",
  digits = 3
)
```

Arguments

`models`	A named list of fitted flexCountReg model objects. This must include 2 or more models.
`coefs`	A logical. The default value 'TRUE' indicates that the coefficients from the models should be included in the table of comparisons.
`AIC`	A logical. The default value 'TRUE' indicates that AIC values for the models should be included.
`BIC`	A logical. The default value 'TRUE' indicates that BIC values for the models should be included.
`RSquare`	A logical. The default value 'TRUE' indicates that the McFadden's Pseudo-R-Squared statistic (comparing against a Poisson regression model) should be included.

tableType	The type of table format to return. Options include "tibble" for returning the table as a tibble, "gt" for a gt table object, or "latex" for a latex table. The default is "tibble".
digits	An integer value indicating the number of decimals to round the table values to.

Value

A table comparing the models supplied to the function in the format specified The table includes coefficients, standard errors, statistical significance, AIC, BIC, and McFadden's Pseudo-R-Squared. Table formats include: "tibble", "gt", and "latex".

Examples

```
# Comparing the NBP model with the NB2 and NB1 models
data("washington_roads")
washington_roads$AADTover10k <- ifelse(washington_roads$AADT>10000,1,0)

nb.1 <- countreg(Total_crashes ~ lnaadt + lnlength + speed50 +
                   ShouldWidth04 + AADTover10k,
                   data=washington_roads, family = 'NB1', method = 'NM')

nb.2 <- countreg(Total_crashes ~ lnaadt + lnlength + speed50 +
                   ShouldWidth04 + AADTover10k,
                   data=washington_roads, family = 'NB2', method = 'NM')


nb.p <- countreg(Total_crashes ~ lnaadt + lnlength + speed50 +
                   ShouldWidth04 + AADTover10k,
                   data=washington_roads, family = 'NBP', method = 'NM')


comptable <-
 regCompTable(list("NB-1"=nb.1, "NB-2"=nb.2, "NB-P"=nb.p), tableType="latex")
print(comptable)
```

---

regCompTest *Compare Regression Models with Likelihood Ratio Test, AIC, and BIC*

Description

This function compares a given regression model to a base model using the Likelihood Ratio (LR) test, Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC).

Usage

```
regCompTest(
  model,
  data = NULL,
  basemodel = "Poisson",
```

```
  variables = FALSE,
  print = FALSE,
  ...
)
```

Arguments

model	A fitted regression model object.
data	An options data frame containing the variables in the model. If not supplied, the original data used to estimate the model will be used.
basemodel	A character string specifying the family of base model to compare against (options include the family from countreg or "Poisson"). Default is "Poisson".
variables	Logical. If TRUE, the base model will include the same variables as the provided model. If FALSE, the base model will be an intercept-only model. Default is FALSE.
print	Logical. If TRUE, a table of the results will be shown. If FALSE, the table of results will not be printed to the console.
...	Additional arguments to be passed to the base model fitting function - options are any argument from the countreg function.

Details

The function performs the following steps:

1. Fits the base model, either a Poisson regression or another specified model.
2. Computes the log-likelihoods of both the provided model and the base model.
3. Calculates the AIC and BIC for both models.
4. Conducts a Likelihood Ratio test to compare the models (if the provided model has more parameters than the base model).
5. Computes McFadden's Pseudo R^2.

The Likelihood-Ratio test is computed as

$$LR = -2(LL_{base\ model} - LL_{model})$$

. The test is chi-squared with degrees of freedom

$$dof = N_{model\ params} - N_{base\ mode\ params}$$

. The AIC is calculated as

$$AIC = -2 \cdot LL + 2 \cdot nparam$$

, and the BIC is calculated as

$$BIC = -2 \cdot LL + nparam \cdot \log(n)$$

.

Value

A list containing the following components:

LL	Log-likelihood of the provided model.
LLbase	Log-likelihood of the base model.
LR	Likelihood Ratio statistic.
LRdof	Degrees of freedom for the Likelihood Ratio test.
AIC	Akaike Information Criterion for the provided model.
AICbase	Akaike Information Criterion for the base model.
BIC	Bayesian Information Criterion for the provided model.
BICbase	Bayesian Information Criterion for the base model.
LR_pvalue	P-value for the Likelihood Ratio test.
PseudoR2	McFadden's Pseudo R^2.
statistics	A tibble format summary of the results.
gtTable	A gt table object summarizing the results.
latexTable	Latex code for a table summarizing the results.
htmlTable	HTML table summarizing the results.

Examples

```
# Comparing the NBP model with the NB2 model
data("washington_roads")
washington_roads$AADTover10k <- ifelse(washington_roads$AADT>10000,1,0)

nbp.base <- countreg(Total_crashes ~ lnaadt + lnlength + speed50 +
                     ShouldWidth04 + AADTover10k,
                     data=washington_roads, family = 'NBP', method = 'NM',
                     max.iters=3000)
regCompTest(nbp.base, washington_roads, basemodel="NB2", print=TRUE)
```

---

renb *Estimate a Random Effects Negative Binomial regression model*

Description

Estimate a Random Effects Negative Binomial regression model

Usage

```
renb(
  formula,
  group_var,
  data,
  method = "NM",
  max.iters = 1000,
  print.level = 0,
  bootstraps = NULL,
  offset = NULL
)
```

Arguments

formula	an R formula.
group_var	the grouping variable(s) for the random effects (e.g., individual ID or other panel ID variables).
data	a dataframe that has all of the variables in the formula.
method	a method to use for optimization in the maximum likelihood estimation. For options, see maxLik. Note that "BHHH" is not available for this function due to the implementation for the random effects.
max.iters	the maximum number of iterations to allow the optimization method to perform.
print.level	Integer specifying the verbosity of output during optimization.
bootstraps	Optional integer specifying the number of bootstrap samples to be used for estimating standard errors. If not specified, no bootstrapping is performed.
offset	an optional offset term provided as a string.

Details

This function estimates a random effects negative binomial (RENB) regression model. This model is based on the NB-1 model. The PDF for the RENB is:

$$f(y_{it}|\mu_{it},a,b) = \frac{\Gamma(a+b)+\Gamma(a+\sum_{t=1}^{n_i}\mu_{it})+\Gamma(b+\sum_{t=1}^{n_i}y_{it})}{\Gamma(a)\Gamma(b)\Gamma(a+b+\sum_{t=1}^{n_i}\mu_{it}+\sum_{t=1}^{n_i}y_{it})}\prod_{t=1}^{n_i}\frac{\Gamma(\mu_{it}+y_{it})}{\Gamma(\mu_{it})\Gamma(y_{it})}$$

Value

An object of class ‘countreg‘ which is a list with the following components:

- model: the fitted model object.
- data: the data frame used to fit the model.
- call: the matched call.
- formula: the formula used to fit the model.

Examples

```
## RENB Model
data("washington_roads")
washington_roads$AADTover10k <-
  ifelse(washington_roads$AADT > 10000, 1, 0) # create a dummy variable
renb.mod <- renb(Animal ~ lnaadt + speed50 + ShouldWidth04 + AADTover10k,
                                data=washington_roads,
                                offset = "lnlength",
                                group_var="ID",
                                method="nm",
                                max.iters = 1000)
summary(renb.mod)
```

`rmse` *Calculate Root Mean Squared Error (RMSE)*

Description

This function calculates the Root Mean Squared Error (RMSE) between observed and predicted values.

Usage

```
rmse(y, mu)
```

Arguments

`y`	Numeric vector representing the observed values.
`mu`	Numeric vector representing the predicted values.

Details

The RMSE is calculated using the formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_i)^2}$$

Where y is the vector of observed values and μ is the vector of predicted values.

Value

Numeric value representing the RMSE.

Examples

```
y <- c(1, 2, 3)
mu <- c(1.1, 1.9, 3.2)
rmse(y, mu)
```

`SichelDistribution` *Sichel Distribution*

Description

Density, distribution function, quantile function, and random generation for the Sichel distribution.

Usage

```
dsichel(x, mu = 1, sigma = 1, gamma = 1, log = FALSE)

psichel(q, mu = 1, sigma = 1, gamma = 1, lower.tail = TRUE, log.p = FALSE)

qsichel(p, mu = 1, sigma = 1, gamma = 1, lower.tail = TRUE, log.p = FALSE)

rsichel(n, mu = 1, sigma = 1, gamma = 1)
```

Arguments

`x`	numeric value or vector of non-negative integer values.
`mu`	numeric; mean of the distribution (mu > 0).
`sigma`	numeric; scale parameter (sigma > 0).
`gamma`	numeric; shape parameter (can be any real number).
`log`, `log.p`	logical; if TRUE, probabilities are given as log(p).
`q`	quantile or vector of quantiles.
`lower.tail`	logical; if TRUE, probabilities are P[X <= x].
`p`	probability or vector of probabilities.
`n`	number of random values to generate.

Details

The Sichel distribution is a three-parameter discrete distribution that generalizes the Poisson-inverse Gaussian distribution. It is useful for modeling overdispersed count data.

The PMF is:

$$f(y|\mu,\sigma,\gamma) = \frac{(\mu/c)^y K_{y+\gamma}(\alpha)}{K_\gamma(1/\sigma)y!(\alpha\sigma)^{y+\gamma}}$$

Value

dsichel gives the density, psichel gives the distribution function, qsichel gives the quantile function, and rsichel generates random deviates.

The length of the result is determined by n for rsichel, and is the maximum of the lengths of the numerical arguments for the other functions.

References

Rigby, R. A., Stasinopoulos, D. M., & Akantziliotou, C. (2008). A framework for modelling overdispersed count data, including the Poisson-shifted generalized inverse Gaussian distribution. Computational Statistics & Data Analysis, 53(2), 381-393.

Examples

```
# Basic usage
dsichel(0:10, mu = 5, sigma = 1, gamma = -0.5)

# Log-probabilities for numerical stability
dsichel(0:10, mu = 5, sigma = 1, gamma = -0.5, log = TRUE)

# CDF
psichel(5, mu = 5, sigma = 1, gamma = -0.5)
```

---

summary.flexCountReg *Custom summary method for flexCountReg models*

Description

Custom summary method for flexCountReg models

Usage

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

Arguments

`object`	A flexCountReg model object.
`...`	Optional parameters that include ‘confint_level‘ and ‘digits‘.

Details

This summary method accounts for bootstrapped or robust standard errors (when used).

Value

Prints the model formula, method used for estimation, number of iterations used, if the model converged, and the log-likelihood. Then, it prints a table containing parameter estimates, standard errors, t-statistics, p-values, and confidence intervals. Also quietly returns a tibble with these values.

Note

Optional parameter ‘confint_level‘: A numeric value between 0 and 1 indicating the confidence level for confidence intervals. Default is 0.95.

Optional parameter ‘digits‘: Number of digits (decimal places) to round to. Default is 3.

Examples

```
# NB2 Model
data("washington_roads")
washington_roads$AADT10kplus <- ifelse(washington_roads$AADT > 10000, 1, 0)
nb2 <- countreg(Total_crashes ~ lnaadt + lnlength + speed50 + AADT10kplus,
                data = washington_roads, family = "NB2",
                dis_param_formula_1 = ~ speed50, method='BFGS')
summary(nb2)
```

---

tidy.flexCountReg *Tidy a flexCountReg object*

Description

Tidy a flexCountReg object

Usage

```
## S3 method for class 'flexCountReg'
tidy(x, ...)
```

Arguments

x	An object of class flexCountReg
...	Additional arguments

Value

A tibble object with columns: term, estimate

Triangular *Triangle Distribution*

Description

These functions provide density, distribution function, quantile function, and random number generation for the Triangle Distribution, specified by its mean, standard deviation, and optional lower and upper bounds.

Usage

```
dtri(x, mode = 0, sigma = 1, upper = NA, lower = NA, log = FALSE)

ptri(
  q,
  mode = 0,
  sigma = 1,
  upper = NA,
  lower = NA,
  lower.tail = TRUE,
  log.p = FALSE
)

qtri(p, mode = 0, sigma = 1, upper = NA, lower = NA)

rtri(n, mode = 0, sigma = 1, upper = NA, lower = NA)
```

Arguments

x	numeric value or a vector of values.
mode	numeric value or vector of mode values for the distribution.
sigma	single value or vector indicating both the positive and negative max differences from the mean (if the difference is the same).
upper	single value or vector for the upper limit of the distribution (must be used with 'lower').
lower	single value or vector for the lower limit of the distribution (must be used with 'upper').
log	logical; if TRUE, probabilities p are given as log(p).
q	quantile or a vector of quantiles.
lower.tail	logical; if TRUE, probabilities p are $P[X \leq x]$ otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).
p	probability or a vector of probabilities.
n	the number of random numbers to generate.

Details

The Triangle Distribution is defined by three points: a (minimum), b (maximum), and c (mode), where the density is zero outside the interval [a, b], increases linearly from a to c, and decreases linearly from c to b.

`dtri` computes the density (PDF) of the Triangle Distribution.

`ptri` computes the CDF of the Triangle Distribution.

`qtri` computes the quantile function of the Triangle Distribution.

`rtri` generates random numbers from the Triangle Distribution.

The mode and standard deviation parameters define the distribution's location and scale, respectively, while the lower and upper bounds explicitly set the minimum and maximum values of the distribution.

Value

dtri gives the density, ptri gives the distribution function, qtri gives the quantile function, and rtri generates random deviates.

The length of the result is determined by n for rtri, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

Examples

```
dtri(4, mode=8, upper=13, lower=1)
ptri(c(0, 1, 2, 3, 5, 7, 9, 10), mode = 3, upper=9, lower = 1)
qtri(c(0.1, 0.3, 0.5, 0.9, 0.95), mode = 3, upper = 9, lower = 1)
rtri(30, mode = 5, sigma = 3)
```

---

washington_roads *Washington Road Crashes*

Description

Crashes on Washington primary roads from 2016, 2017, and 2018. Data acquired from Washington Department of Transportation through the Highway Safety Information System (HSIS).

Usage

```
washington_roads
```

Format

Data frame compiled from roadway, traffic, and police-reported crash data that has 1,501 rows and 13 columns:

ID Anonymized road ID. Factor.

Year Year. Integer.

AADT Annual Average Daily Traffic (AADT). Double.

Length Segment length in miles. Double.

Total_crashes Total crashes. Integer.

lnaadt Natural logarithm of AADT. Double.

lnlength Natural logarithm of length in miles. Double.

speed50 Indicator of whether the speed limit is 50 mph or greater. Binary.

ShouldWidth04 Indicator of whether the shoulder is 4 feet or wider. Binary.

Fatal_crashes Total number of non-intersection fatal crashes for the road segment

Injury_crashes Total number of non-intersection Injury crashes for the road segment

Animal Total number of non-intersection animal-related crashes for the road segment

Rollover Total number of non-intersection rollover crashes for the road segment

Source

https://highways.dot.gov/research/safety/hsis

Index

∗ **datasets**
PoissonLindley, 46
washington_roads, 65

COMDistribution, 3
cor2cov, 5
corr_haltons, 5
countreg, 7, *20*, *57*
countreg.rp, 20
cure_plot, *23*
cureplot, 22

dcom *(*COMDistribution*)*, 3
dgwar *(*Generalized-Waring*)*, 26
dinvgamma *(*invgamma*)*, 29
dlindley, 24
dpge *(*PoissonGeneralizedExponential*)*, 38
dpinvgamma *(*PoissonInverseGamma*)*, 41
dpinvgaus *(*PoissonInverseGaussian*)*, 42
dplind *(*PoissonLindley*)*, 46
dplindGamma, *16*
dplindGamma *(*NegativeBinomialLindley*)*, 35
dplindLnorm, *16*
dplindLnorm *(*PoissonLindleyLognormal*)*, 47
dpLnorm *(*PoissonLognormal*)*, 49
dpoisweibull *(*PoissonWeibull*)*, 51
dsichel *(*SichelDistribution*)*, 61
dtri *(*Triangular*)*, 63

flexCountReg-class, 25

Generalized-Waring, 26
gt, *56*, *58*

halton_dists, 28

invgamma, 29

mae, 31
maxLik, *37*, *59*
mgf_lognormal, 32
msg1 *(*PoissonLindley*)*, 46
myAIC, 33
myBIC, 34

NegativeBinomialLindley, 35

pcom *(*COMDistribution*)*, 3
pgwar *(*Generalized-Waring*)*, 26
pinvgamma *(*invgamma*)*, 29
plindley *(*dlindley*)*, 24
poisLind.re *(*poisLindRE*)*, 37
poisLindRE, 37
PoissonGeneralizedExponential, 38
PoissonInverseGamma, 41
PoissonInverseGaussian, 42
PoissonLindley, 46
PoissonLindleyLognormal, 47
PoissonLognormal, 49
PoissonWeibull, 51
ppge *(*PoissonGeneralizedExponential*)*, 38
ppinvgamma *(*PoissonInverseGamma*)*, 41
ppinvgaus *(*PoissonInverseGaussian*)*, 42
pplind *(*PoissonLindley*)*, 46
pplindGamma *(*NegativeBinomialLindley*)*, 35
pplindLnorm *(*PoissonLindleyLognormal*)*, 47
ppLnorm *(*PoissonLognormal*)*, 49
ppoisweibull *(*PoissonWeibull*)*, 51
predict.flexCountReg, *23*, 53
psichel *(*SichelDistribution*)*, 61
ptri *(*Triangular*)*, 63

qcom *(*COMDistribution*)*, 3
qgwar *(*Generalized-Waring*)*, 26
qinvgamma *(*invgamma*)*, 29
qlindley *(*dlindley*)*, 24
qpge *(*PoissonGeneralizedExponential*)*, 38

qpinvgamma *(*PoissonInverseGamma*)*, 41
qpinvgaus *(*PoissonInverseGaussian*)*, 42
qplind *(*PoissonLindley*)*, 46
qplindGamma *(*NegativeBinomialLindley*)*, 35
qplindLnorm *(*PoissonLindleyLognormal*)*, 47
qpLnorm *(*PoissonLognormal*)*, 49
qpoisweibull *(*PoissonWeibull*)*, 51
qsichel *(*SichelDistribution*)*, 61
qtri *(*Triangular*)*, 63

rcom *(*COMDistribution*)*, 3
regCompTable, 55
regCompTest, 56
renb, 58
rgwar *(*Generalized-Waring*)*, 26
rinvgamma *(*invgamma*)*, 29
rlindley *(*dlindley*)*, 24
rmse, 60
rpge *(*PoissonGeneralizedExponential*)*, 38
rpinvgamma *(*PoissonInverseGamma*)*, 41
rpinvgaus *(*PoissonInverseGaussian*)*, 42
rplind *(*PoissonLindley*)*, 46
rplindGamma *(*NegativeBinomialLindley*)*, 35
rplindLnorm *(*PoissonLindleyLognormal*)*, 47
rpLnorm *(*PoissonLognormal*)*, 49
rpoisweibull *(*PoissonWeibull*)*, 51
rsichel *(*SichelDistribution*)*, 61
rtri *(*Triangular*)*, 63

SichelDistribution, 61
summary.flexCountReg, 62

tidy.flexCountReg, 63
Triangular, 63

washington_roads, 65