Package 'countsFun'

November 7, 2025

Type Package

Title Fit count time series models

Version 0.5

Maintainer Stefanos Kechagias <stefanoskeh@gmail.com>

Description The lgc package fits Latent Gaussian Count time series models using the copula transformations and simulation inference techniques developed by Jia et al. (2023)

License GPL (>= 3)

Encoding UTF-8

LazyData true

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports doParallel, itsmr, iZID, mixtools, optextras, extraDistr, optimx, parallel, foreach, VGAM, MASS, truncnorm

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 3.5)

References Jia, Y., Kechagias, S., Livsey, J., Lund, R. and Pipiras, V., 2023. Latent Gaussian count time series. Journal of the American Statistical Association, 118(541), pp.596-606.

Contents

`countsFun-package` *countsFun: Fit count time series models*

Description

The lgc package fits Latent Gaussian Count time series models using the copula transformations and simulation inference techniques developed by Jia et al. (2023)

Author(s)

Maintainer: Stefanos Kechagias <stefanoskeh@gmail.com>

Authors:

- James Livsey <james.a.livsey@gmail.com>
- Robert Lund <lund@clemson.edu>
- Vladas Pipiras <pipiras@email.unc.edu>

AIC.lgc *AIC for lgc Model*

Description

Returns the AIC value for a fitted lgc model.

Usage

```
## S3 method for class 'lgc'
AIC(object, ...)
```

Arguments

object	An object of class lgc.
...	Additional arguments (currently unused).

Value

Numeric. AIC value.

BIC.lgc *BIC for lgc Model*

Description

Returns the BIC value for a fitted lgc model.

Usage

```
## S3 method for class 'lgc'
BIC(object, ...)
```

Arguments

object	An object of class lgc.
...	Additional arguments (currently unused).

Value

Numeric. BIC value.

`CheckStability` *Check Causality and Invertibility of ARMA Model*

Description

Checks whether the specified ARMA model is stable — i.e., whether the AR polynomial is causal and the MA polynomial is invertible. This is done by examining the location of the roots of the characteristic polynomials.

Usage

```
CheckStability(AR, MA)
```

Arguments

`AR`	Numeric vector. Autoregressive (AR) coefficients of the model. Can be `NULL` if the model has no AR part.
`MA`	Numeric vector. Moving average (MA) coefficients of the model. Can be `NULL` if the model has no MA part.

Details

The function uses `polyroot` to compute the roots of the AR and MA characteristic polynomials:

- The AR polynomial is defined as $1 - AR_1 z - AR_2 z^2 - \ldots$.
- The MA polynomial is defined similarly.

If any root lies **within** the unit circle (i.e., has modulus < 1), the model is considered unstable.

Value

Integer. Returns `1` if the model is **not stable** (i.e., at least one root of the AR or MA polynomial lies inside the unit circle), and `0` if the model is stable.

Examples

```
# Stable ARMA(1,1)
CheckStability(AR = 0.5, MA = 0.3)  # returns 0

# Unstable ARMA(1,1)
CheckStability(AR = 1.2, MA = 0.3)  # returns 1

# AR-only model
CheckStability(AR = c(0.9), MA = NULL)

# MA-only model
CheckStability(AR = NULL, MA = c(1.5))
```

coefficients.lgc *Coefficients for lgc Model*

Description

Returns the estimated coefficients from a fitted `lgc` model.

Usage

```
## S3 method for class 'lgc'
coefficients(object, ...)
```

Arguments

`object`	An object of class `lgc`.
`...`	Additional arguments (currently unused).

Value

Matrix of parameter estimates.

ComputeResiduals *Computes residuals from a fitted latent Gaussian count (LGC) model using the approach described in relation (41) of Jia et al. (2021).*

Description

Computes residuals from a fitted latent Gaussian count (LGC) model using the approach described in relation (41) of Jia et al. (2021).

Usage

```
ComputeResiduals(lgc)
```

Arguments

`lgc`	A list object containing the outcome of a model fit, typically returned by the package's main wrapper function `lgc`.

Details

This function computes residuals as defined in relation (41) of Jia et al. (2021)

Value

A numeric vector of residuals, computed via the inverse-normal transformation of the cumulative distribution function (CDF) of the observed counts.

References

Jia, Y., Kechagias, S., Livsey, J., Lund, R., & Pipiras, V. (2021). Latent Gaussian Count Time Series. *Journal of the American Statistical Association*, 118(541), 596–606. doi:10.1080/01621459.2021.1944874

Criteria.lgc *Compute AIC, BIC, and AICc for Model Evaluation*

Description

Computes the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and corrected AIC (AICc) for a fitted model, based on the log-likelihood, sample size, and number of parameters.

Usage

```
Criteria.lgc(loglik, mod)
```

Arguments

loglik	Numeric. The log-likelihood value of the fitted model.
mod	A list containing model-related metadata, typically created by ModelScheme. It must include: • nparms: Number of estimated parameters. • n: Sample size. • EstMethod: Estimation method used ("GL" or "PFR").

Value

A numeric vector of length 3 containing the values:

AIC Akaike Information Criterion

BIC Bayesian Information Criterion

AICc Corrected Akaike Information Criterion

Examples

```
mod <- list(nparms = 5, n = 100, EstMethod = "PFR")
loglik <- -123.45
Criteria.lgc(loglik, mod)
```

dGpois *Generalized Poisson PMF*

Description

Computes the probability mass function (PMF) of the Generalized Poisson distribution using the mean-parametrization (Famoye 1994, relation 2.4).

Usage

```
dGpois(y, a, m, log = FALSE)
```

Arguments

y	Non-negative integer vector of quantiles.
a	Dispersion parameter.
m	Mean parameter.
log	Logical; if TRUE, probabilities are returned on the log scale.

Value

A numeric vector of (log-)probabilities.

---

dmixpois1 *Mixture of Two Poisson Distributions (PMF)*

Description

Computes the probability mass function of a two-component mixture of Poisson distributions.

Usage

```
dmixpois1(x, lam1, lam2, p, log = FALSE)
```

Arguments

x	Non-negative integer(s). Vector of quantiles.
lam1	Mean of the first Poisson component (must be > 0).
lam2	Mean of the second Poisson component (must be > 0).
p	Mixing probability for the first component (0 <= p <= 1).
log	Logical; if TRUE, probabilities p are given as log(p).

Value

Numeric vector of mixture probabilities (or log-probabilities).

Examples

```
# Probability of observing exactly 3 under the mixture
dmixpois1(3, lam1 = 2, lam2 = 5, p = 0.4)

# Log-probability
dmixpois1(3, lam1 = 2, lam2 = 5, p = 0.4, log = TRUE)

# Vectorized input
dmixpois1(0:5, lam1 = 2, lam2 = 5, p = 0.4)
```

drinksales *Example sales dataset*

Description

A synthetic example dataset for demonstration or testing purposes.

Usage

```
drinksales
```

Format

A data frame with N rows and M columns (adjust as needed).

Source

Dominick supermarket chain

FitMultiplePF_Res *Optimization wrapper to fit PF likelihood with resampling (new version)*

Description

Fits the particle filter log-likelihood using resampling. This version expects a complete model object (created by ModelScheme) and performs one optimization per value of mod$ParticleNumber.

Usage

```
FitMultiplePF_Res(theta, mod)
```

Arguments

theta	Numeric vector. Initial parameter values.
mod	A list containing model-related metadata and control settings, typically returned by ModelScheme.

Value

A list containing parameter estimates, standard errors, log-likelihood, AIC, BIC, AICc, and optimization diagnostics.

Examples

```
# Specify model characteristics
CountDist      <- "Poisson"
n              <- 100
ARMAModel      <- c(1, 0)
MargParm       <- 3
ARParm         <- 0.5
MAParm         <- NULL
initialParam   <- c(3.2, 0.9 * ARParm, 0.8 * MAParm)
Task           <- "Optimization"

# Simulate data
set.seed(2)
DependentVar <- sim_lgc(n, CountDist, MargParm, ARParm, MAParm)

# Prepare the model object
mod <- ModelScheme(
  DependentVar   = DependentVar,
  CountDist      = CountDist,
  ARMAModel      = ARMAModel,
  initialParam   = initialParam,
  Task           = Task
)

# Fit the model
fit <- FitMultiplePF_Res(initialParam, mod)
fit
```

`InitialEstimates` *Compute Initial Parameter Estimates*

Description

Generates initial parameter estimates based on the model specifications provided in `mod`. These starting values are typically used as input to optimization routines or simulations in count time series models with ARMA dependencies and various marginal count distributions.

Usage

```
InitialEstimates(mod)
```

Arguments

`mod`	List. A list containing all model specifications, including:
	DependentVar Numeric. The dependent time series variable to be modeled.
	Task Character. The task requested by the user (e.g., Evaluation, Optimization, Synthesis, etc.).
	ParticleNumber Integer. Number of particles to use.
	Regressor Optional independent variable(s).
	CountDist Character string. Specifies the count marginal distribution.

Details

The initial estimates are heuristically determined and may depend on the distribution type, sample statistics (e.g., mean, variance), or defaults chosen for stability. These estimates are used as starting points in numerical optimization procedures. GLM and MoM estimates are used for marginal parameters and Yulew-Walker for ARMA.

Value

A numeric vector of initial parameter estimates consistent with the model structure described in `mod`. The vector includes:

- Marginal distribution parameters
- Regression coefficients (if any)
- AR and MA coefficients

The names of the vector elements (if assigned) match those in `mod$parmnames`.

Examples

```
mod <- ModelScheme(
  DependentVar = rpois(100, lambda = 3),
  CountDist = "Poisson",
  ARMAModel = c(1, 1),
  Intercept = TRUE
)
init <- InitialEstimates(mod)
print(init)
```

`InnovAlg` *Innovations Algorithm for Time Series Prediction*

Description

Applies the Innovations Algorithm to compute the optimal one-step-ahead predictors and associated prediction error variances based on the autocovariance sequence of a stationary time series. The algorithm uses the ARMA parameters provided in `Parms` and the autocovariance vector `gamma` to generate recursive estimates of innovation coefficients and variances.

Usage

```
InnovAlg(Parms, gamma, mod)
```

Arguments

`Parms`	A list containing the ARMA model parameters: • `AR`: Vector of autoregressive (AR) coefficients. • `MA`: Vector of moving average (MA) coefficients.
`gamma`	Numeric vector. The autocovariance sequence of the observed time series. Typically length N, where N is the time series length.
`mod`	List. A list containing all model specifications, including:

DependentVar Numeric. The dependent time series variable to be modeled.

Task Character. The task requested by the user (e.g., Evaluation, Optimization, Synthesis, etc.).

ParticleNumber Integer. Number of particles to use.

Regressor Optional independent variable(s).

CountDist Character string. Specifies the count marginal distribution.

Details

The recursion is terminated based on a fixed tolerance (`mod$maxdiff`) for variance convergence in MA models, or after a fixed number of steps in AR-only models. The `kappa(i,j)` function computes the autocovariance terms needed for recursion using model coefficients and the autocovariance vector `gamma`.

Value

A list with the following elements:

`n` Number of recursion steps performed.

`thetas` List of innovation coefficients (one per step). Each element is a vector of length `q`.

`v` Vector of innovation variances at each step (normalized by `v[1]`).

Examples

```
# ARMA(1,1) example
gamma <- ARMAacf(ar = 0.5, ma = 0.4, lag.max = 20)
mod <- list(nAR = 1, nMA = 1, maxdiff = 1e-8)
Parms <- list(AR = 0.5, MA = 0.4)
result <- InnovAlg(Parms, gamma, mod)
str(result)
```

---

lgc *Main Wrapper for Fitting Latent Gaussian Count Time Series Models*

Description

Fits latent Gaussian count models using particle filtering methods for various count distributions (e.g., Poisson, Negative Binomial, Zero-Inflated Poisson) and optional ARMA dependencies and regressors. The function supports tasks such as model evaluation, optimization, and simulation.

Usage

```
lgc(
  formula = NULL,
  data = NULL,
  EstMethod = "PFR",
  CountDist = NULL,
  ARMAModel = NULL,
  ParticleNumber = 5,
  epsilon = 0.5,
```

```
  initialParam = NULL,
  TrueParam = NULL,
  Task = "Evaluation",
  SampleSize = NULL,
  nsim = NULL,
  no_cores = 1,
  OptMethod = "L-BFGS-B",
  OutputType = "list",
  ParamScheme = NULL,
  maxdiff = 10^(-8),
  ntrials = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

formula	An object of class formula (e.g., y ~ x1 + x2). Specifies the response and regressors.
data	A data frame containing the variables used in the formula.
EstMethod	Character. Estimation method to use. Default is "PFR" (particle filter with resampling).
CountDist	Character. Distribution of the count variable. Supported: "Poisson", "Negative Binomial", "ZIP", etc.
ARMAModel	Numeric vector of length 2 specifying AR and MA orders (e.g., c(1,1) for ARMA(1,1)).
ParticleNumber	Number of particles to use in the particle filter.
epsilon	Numeric. Smoothing parameter for the resampling step (if applicable).
initialParam	Optional. Numeric vector of initial parameter values. If NULL, estimates are generated internally.
TrueParam	Optional. True parameter vector used when Task == "Simulation".
Task	Character. Specifies the task to perform: "Evaluation", "Optimization", or "Simulation".
SampleSize	Integer. Used only when Task == "Simulation".
nsim	Integer. Number of simulations to run (if Task == "Simulation").
no_cores	Number of CPU cores to use in parallel computing (only for simulation).
OptMethod	Optimization method passed to optimx (e.g., "L-BFGS-B").
OutputType	Output format: "list" (default) or "wide".
ParamScheme	Optional. Parameterization scheme name (if applicable).
maxdiff	Numeric. Convergence threshold for the Innovations algorithm.
ntrials	Integer. Required if CountDist == "Binomial" to specify number of trials.
verbose	Logical. If TRUE (default), informative messages are printed during execution. Set to FALSE to suppress messages.
...	Additional arguments (currently unused).

Details

This function serves as the main interface for the `lgc` package. It parses inputs, prepares model specifications, fits the model via particle filtering with optional optimization, and computes residuals using results from Jia et al. (2021).

Value

If `Task` is `"Evaluation"` or `"Optimization"`, returns a list of class `"lgc"` containing:

- `ParamEstimates`, `StdErrors`, `FitStatistics`, `OptimOutput`
- `residuals`: Model residuals
- `Model`, `Task`, `EstMethod`, `SampleSize`, etc.

If `Task == "Simulation"`, returns a list of `lgc` objects (length = `nsim`).

References

Jia, Y., Kechagias, S., Livsey, J., Lund, R., & Pipiras, V. (2021). Latent Gaussian Count Time Series. *Journal of the American Statistical Association*, 118(541), 596–606. doi:10.1080/01621459.2021.1944874

Examples

```
CountDist      = "Generalized Poisson"
alpha          = 1
b0             = 0.5
b1             = 2
MargParm       = c(b0,b1,alpha)
ARParm         = 0.75
MAParm         = NULL
ARMAModel      = c(length(ARParm),length(MAParm))
SampleSize     = 50
Regressor      = rbinom(SampleSize,1,0.1)
Intercept      = TRUE

# simulate data with the old lgc function
set.seed(1)
DependentVar  = sim_lgc(SampleSize, CountDist, MargParm, ARParm, MAParm, Regressor,Intercept)

# save the data in a data frame
df = data.frame(DependentVar, Regressor)

# specify the regression model
formula = DependentVar~Regressor

# specify task
Task = "Evaluation"

# specify parameters to evaluate the log-likelihood
initialParam = c(MargParm, ARParm, MAParm)

# call the wrapper function with less arguments
mylgc = lgc(formula   = formula,
            data      = df,
            CountDist = CountDist,
            ARMAModel = ARMAModel,
```

```
            Task = Task,
             initialParam = initialParam)

# check the results
summary(mylgc)
```

`logLik.lgc` *Log-Likelihood for lgc Model*

Description

Returns the log-likelihood value stored in a fitted `lgc` model object.

Usage

```
## S3 method for class 'lgc'
logLik(object, ...)
```

Arguments

`object`	An object of class `lgc`.
`...`	Additional arguments (currently unused).

Value

Numeric. Log-likelihood value.

`model.lgc` *Model Summary for lgc Object*

Description

Returns a data frame summarizing the distribution and ARMA model structure.

Usage

```
## S3 method for class 'lgc'
model(object, ...)
```

Arguments

`object`	An object of class `lgc`.
`...`	Additional arguments (currently unused).

Value

A data frame with two columns: distribution and model type.

`ModelScheme` *Create and Validate Model Specification for Particle Filtering*

Description

Parses and validates the full model specification and specified options. This function gathers model components (data, distribution, dependence structure, optimization parameters, etc.) and returns a structured list used as input to other core functions in the package.

Usage

```
ModelScheme(
  DependentVar = NULL,
  Regressor = NULL,
  Intercept = NULL,
  EstMethod = "PFR",
  ARMAModel = c(0, 0),
  CountDist = "NULL",
  ParticleNumber = 5,
  epsilon = 0.5,
  initialParam = NULL,
  TrueParam = NULL,
  Task = "Evaluation",
  SampleSize = NULL,
  OptMethod = "L-BFGS-B",
  OutputType = "list",
  ParamScheme = 1,
  maxdiff = 10^(-8),
  ntrials = NULL,
  verbose = TRUE,
  nsim = NULL,
  ...
)
```

Arguments

`DependentVar`	Numeric vector or data frame. The response (count) time series.
`Regressor`	Optional matrix or data frame of covariates.
`Intercept`	Logical. Whether to include an intercept in the model.
`EstMethod`	Character. Estimation method. Currently only `"PFR"` is supported.
`ARMAModel`	Integer vector of length 2. Specifies the order of the AR and MA components.
`CountDist`	Character. The name of the count distribution (e.g., "Poisson", "Negative Binomial", "ZIP").
`ParticleNumber`	Integer. Number of particles to be used in the filter.
`epsilon`	Numeric. Resampling occurs when the effective sample size (ESS) drops below `epsilon * N`.
`initialParam`	Numeric vector. Initial parameter values for optimization.
`TrueParam`	Numeric vector. True parameter values (used in simulation/synthesis, ignored otherwise).

Task	Character. One of "Evaluation", "Optimization", "Simulation", or "Synthesis".
SampleSize	Integer. Required when Task == "Synthesis".
OptMethod	Character. Optimization method, e.g., "L-BFGS-B".
OutputType	Character. Type of output, typically "list".
ParamScheme	Integer. Indicates the parameter configuration scheme.
maxdiff	Numeric. Convergence criterion for optimization.
ntrials	Integer. Number of trials (used in Binomial model).
verbose	Logical. If TRUE (default), informative messages are printed during execution.
nsim	Integer. Number of replications. Required when Task == "Simulation". Set to FALSE to suppress messages.
...	Additional arguments (currently unused).

Details

This function validates the inputs (e.g., distribution support, ARMA structure, initial parameter lengths), computes the number of model parameters, constructs marginal distribution functions based on the count distribution, and configures parameter bounds. It is a required pre-processing step before fitting or simulating count time series models.

Value

A named list containing model specifications, validated inputs, marginal distribution functions (PDF, CDF, inverse CDF), initial and true parameters, parameter bounds, parameter names, and utility values for likelihood evaluation.

---

nobs.lgc *Number of Observations in lgc Model*

Description

Returns the number of observations used in fitting the model.

Usage

```
## S3 method for class 'lgc'
nobs(object, ...)
```

Arguments

object	An object of class lgc.
...	Additional arguments (currently unused).

Value

An integer, the number of observations.

parse_formula *Parse Formula into Response and Regressors*

Description

Parses a standard R formula and extracts the response variable, predictor variables, and intercept status.

Usage

```
parse_formula(formula)
```

Arguments

formula	An object of class formula, typically in the form y ~ x1 + x2.

Details

This function is useful for preprocessing model formulas in custom modeling functions. It uses terms to extract structure from the formula object.

Value

A list with three elements:

DependentVar Character string of the response variable name.

Regressor Character vector of predictor variable names, or NULL if none.

intercept Logical, TRUE if an intercept is included, FALSE otherwise.

See Also

terms, model.frame

ParticleFilter *Particle Filter Latent Gaussian Count Models*

Description

Implements a particle filtering algorithm for latent Gaussian count time series models with potentially misspecified ARMA dependence structures. The algorithm sequentially updates latent Gaussian states, particle weights, and the log-likelihood, using the Innovations Algorithm to obtain one-step-ahead predictors and innovation variances. Corrections are applied to truncated normal sampling limits and particle weights when numerical instability or boundary issues occur.

Usage

```
ParticleFilter(theta, mod)
```

Arguments

`theta`	Numeric vector. Current parameter estimates including marginal, AR, and MA parameters.
`mod`	List. Model specification object containing all necessary elements for filtering, such as:

`DependentVar` Numeric vector of observed counts.

`CountDist` Character string specifying the count distribution (e.g., `"Poisson"`, `"Binomial"`).

`ARMAModel` Numeric vector `c(p, q)` specifying AR and MA orders.

`ParticleNumber` Integer. Number of particles used in the filter.

`n` Integer. Length of the observed time series.

`mycdf` User-supplied function returning the marginal CDF of the count distribution.

`mypdf` User-supplied function returning the marginal PDF of the count distribution.

`Regressor` Optional numeric vector or matrix of regressors.

`nreg` Integer. Number of regressors.

`verbose` Logical. If `TRUE`, prints diagnostic messages during filtering.

`maxdiff` Numeric tolerance used in the Innovations Algorithm convergence criterion.

Details

The function evaluates the (negative) log-likelihood for an LGC model under ARMA dynamics.

Value

Returns the numeric value of the negative log-likelihood ($-\ell(\theta)$). If numerical issues occur (e.g., unstable ARMA coefficients, degenerate weights, or non-finite log-likelihood), the function returns a large penalty value (10^8) to guide optimization away from invalid parameter regions.

`ParticleFilter_Res_ARMA` *Particle Filter likelihood function with Resampling for ARMA Models*

Description

Uses particle filtering with resampling to approximate the likelihood of a specified count time series model with an underlying ARMA dependence structure.

Usage

```
ParticleFilter_Res_ARMA(theta, mod)
```

Arguments

theta	Numeric vector. Parameter vector for the model.
mod	List. A list containing all model specifications, including:

DependentVar Numeric. The dependent time series variable to be modeled.

Task Character. The task requested by the user (e.g., Evaluation, Optimization, Synthesis, etc.).

ParticleNumber Integer. Number of particles to use.

Regressor Optional independent variable(s).

CountDist Character string. Specifies the count marginal distribution.

Value

Numeric. Approximate log-likelihood of the model.

Examples

```
# Example: Compute the log-likelihood of a Poisson-ARMA(1,0) model
CountDist      <- "Poisson"
n              <- 100
ARMAModel      <- c(1,0)
MargParm       <- 3
ARParm         <- 0.5
MAParm         <- NULL
initialParam   <- c(3, ARParm, MAParm)

# Simulate data
set.seed(2)
DependentVar <- sim_lgc(n, CountDist, MargParm, ARParm, MAParm)

# Build model specification
mod <- ModelScheme(
  DependentVar   = DependentVar,
  CountDist      = CountDist,
  ARMAModel      = ARMAModel,
  initialParam   = initialParam
)

# Compute log-likelihood at a given parameter point
ll <- ParticleFilter_Res_ARMA(initialParam, mod)
ll
```

PDvalues *Modifies the standard particle filtering procedure to return the predictive distribution at each time point of a latent Gaussian count time series model, instead of just the log-likelihood.*

Description

Modifies the standard particle filtering procedure to return the predictive distribution at each time point of a latent Gaussian count time series model, instead of just the log-likelihood.

Usage

```
PDvalues(theta, mod)
```

Arguments

theta	Numeric vector. Parameter vector containing marginal and ARMA parameters.
mod	List. A list containing all model specifications, including:
	DependentVar Numeric. The dependent time series variable to be modeled.
	Task Character. The task requested by the user (e.g., Evaluation, Optimization, Synthesis, etc.).
	ParticleNumber Integer. Number of particles to use.
	Regressor Optional independent variable(s).
	CountDist Character string. Specifies the count marginal distribution.

Details

This function runs a particle filter with resampling to estimate the predictive distribution of the latent state at each time point. It leverages the Innovations Algorithm to calculate ARMA-based forecasts of the latent process and approximates conditional distributions using truncated normal draws. See Section 3.4 in Jia et al. (2021)

Value

A numeric matrix of dimension `2 x n`, where each column corresponds to a time point:

- First row: lower tail probability for the observed count
- Second row: predictive probability mass at `Y_t + 1`

References

Jia, Y., Kechagias, S., Livsey, J., Lund, R., & Pipiras, V. (2021). Latent Gaussian Count Time Series. *Journal of the American Statistical Association*, 118(541), 596–606. doi:10.1080/01621459.2021.1944874

See Also

ComputeWeights, SampleTruncNormParticles, InnovAlg

Examples

```
n             = 10
Regressor     = data.frame(runif(n),runif(n))
Intercept     = TRUE
ARMAModel     = c(2,0)
ARParm        = c(0.5, 0.2)
MAParm        = NULL
CountDist     = "Poisson"
b0            = 1
b1            = 4
b2            = 2
MargParm      = c(b0,b1,b2)

# simulate data
set.seed(2)
```

```
DependentVar   = sim_lgc(n, CountDist, MargParm, ARParm, MAParm, Regressor, Intercept)

mod = ModelScheme(DependentVar   = DependentVar,
                  Regressor      = Regressor,
                  Intercept      = Intercept,
                  CountDist      = CountDist,
                 ARMAModel      = ARMAModel)

# select a parameter point
theta <- c(MargParm, ARParm, MAParm)

# compute the Predictive distribution
PDvalues(theta, mod)
```

---

pGpois *Generalized Poisson CDF*

Description

Computes the cumulative distribution function (CDF) of the Generalized Poisson distribution.

Usage

```
pGpois(x, a, m, lower.tail = TRUE, log.p = FALSE)
```

Arguments

x	Integer or numeric vector. Values at which to evaluate the CDF.
a	Numeric. Dispersion parameter.
m	Numeric scalar or vector. Mean parameter(s).
lower.tail	Logical; if TRUE (default), probabilities are $P(X \leq x)$. Otherwise, $P(X > x)$.
log.p	Logical; if TRUE, probabilities are returned on the log scale.

Value

A numeric vector or matrix of cumulative probabilities.

See Also

dGpois

Examples

```
pGpois(3, 0.4, 2)
pGpois(3, 0.4, 2, lower.tail = FALSE, log.p = TRUE)
```

`PITvalues` *Compute Probability Integral Transform (PIT) Histogram Values*

Description

Computes histogram values for the Probability Integral Transform (PIT) as described in relations (39)–(40) of the JASA paper on latent Gaussian count time series modeling. This function processes predictive distributions and outputs the PIT histogram bins for calibration assessment.

Usage

```
PITvalues(H, predDist)
```

Arguments

`H`	Integer. Number of histogram bins to divide the [0,1] PIT range.
`predDist`	A numeric matrix of dimension `2 x n`, typically returned by `PDvalues`. Each column corresponds to a time point, with: • First row: lower tail probability of the observed count • Second row: cumulative probability including the observed count

Details

The PIT is used to assess the calibration of probabilistic forecasts. For discrete data, the PIT is computed using randomized methods or as a piecewise function, as detailed in the latent Gaussian count models literature.

This implementation applies the formulas:

$$PIT(Y_t) \sim U(0,1)$$

if predictive distributions are correctly specified.

Value

A numeric vector of length `H` representing the PIT histogram values (bin heights). These can be plotted to assess calibration of predictive distributions.

References

Jia, Y., Kechagias, S., Livsey, J., Lund, R., & Pipiras, V. (2021). Latent Gaussian Count Time Series. *Journal of the American Statistical Association*, 118(541), 596–606. doi:10.1080/01621459.2021.1944874

See Also

`PDvalues`, `ComputeResiduals`

Examples

```
## Not run:
predDist <- PDvalues(theta, mod)
hist(PITvalues(10, predDist), breaks = 10, main = "PIT Histogram")

## End(Not run)
```

pmixpois1 *Mixture of Two Poisson Distributions (CDF)*

Description

Computes the cumulative distribution function of a two-component mixture of Poisson distributions.

Usage

```
pmixpois1(x, lam1, lam2, p, lower.tail = TRUE, log.p = FALSE)
```

Arguments

x	Non-negative integer(s). Vector of quantiles.
lam1	Mean of the first Poisson component (must be > 0).
lam2	Mean of the second Poisson component (must be > 0).
p	Mixing probability for the first component (0 <= p <= 1).
lower.tail	Logical; if TRUE (default), probabilities are $P(X \leq x)$. Otherwise, $P(X > x)$.
log.p	Logical; if TRUE, probabilities p are given as log(p).

Value

Numeric vector of mixture CDF values (or log-probabilities).

Examples

```
# CDF at 3 under the mixture
pmixpois1(3, lam1 = 2, lam2 = 5, p = 0.4)

# Upper tail probability in log scale
pmixpois1(3, lam1 = 2, lam2 = 5, p = 0.4, lower.tail = FALSE, log.p = TRUE)
```

print.lgc *Print a Latent Gaussian Count (LGC) Model*

Description

Nicely formats and prints a fitted lgc model object, including model type, estimation details, fit statistics, and parameter estimates. Designed to provide a concise summary suitable for quick inspection by statisticians and time-series practitioners.

Usage

```
## S3 method for class 'lgc'
print(x, digits = 4, ...)
```

Arguments

x	An object of class "lgc".
digits	Integer; number of significant digits to display (default = 4).
...	Additional arguments (currently unused).

Details

This method provides a human-readable printout of an LGC model object. For more comprehensive model information, including diagnostic metrics and statistical inference, use summary.lgc().

Value

Invisibly returns the input object x.

See Also

summary.lgc for a detailed summary method.

Examples

```
## Not run:
  # Fit an example LGC model
  mylgc <- lgc(formula = y ~ x, data = simdata, EstMethod = "PFR")

  # Print the model summary
  print(mylgc)

## End(Not run)
```

---

qGpois *Generalized Poisson Quantile Function (vectorized and scalar)*

Description

Computes the quantile function (inverse CDF) of the Generalized Poisson distribution for one or more values of the mean parameter m.

Usage

```
qGpois(p, a, m)
```

Arguments

p	Numeric vector of probabilities (between 0 and 1).
a	Numeric. Dispersion parameter.
m	Numeric scalar or vector. Mean parameter(s).

Value

A numeric vector or matrix of quantiles.

See Also

pGpois

Examples

```
qGpois(0.8,0.4,3)
```

---

qmixpois1 *Vectorized Quantile Function for Mixed Poisson Distribution*

Description

Computes quantiles of the mixed Poisson distribution using a vectorized approach with `mapply()`.

Usage

```
qmixpois1(p, lam1, lam2, prob)
```

Arguments

p	Numeric vector of probabilities (between 0 and 1).
lam1	Numeric or vector. Mean(s) of the first Poisson component.
lam2	Numeric or vector. Mean(s) of the second Poisson component.
prob	Numeric between 0 and 1. Mixing probability for the first component.

Details

This function is a vectorized version of qmixpois1 using `mapply()` to compute quantiles efficiently.

Value

A numeric vector of quantiles corresponding to p.

Examples

```
qmixpois1(c(0.1, 0.5, 0.9), lam1 = 2, lam2 = 5, prob = 0.6)
```

residuals.lgc *Extract Residuals from an lgc Model*

Description

Computes residuals from a fitted lgc model using the ARMA structure specified in the model.

Usage

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

Arguments

object	An object of class lgc.
...	Additional arguments (currently unused).

Value

A numeric vector of residuals of length nobs(object).

Examples

```
## Not run:
  residuals(mylgc)

## End(Not run)
```

se.lgc *Standard Errors for lgc Model*

Description

Returns the standard errors from a fitted lgc model.

Usage

```
## S3 method for class 'lgc'
se(object, ...)
```

Arguments

object	An object of class lgc.
...	Additional arguments (currently unused).

Value

Matrix of standard errors.

SimPois2_AR1_075_N1k *Simulated Poisson AR(1) Time Series*

Description

A simulated Poisson time series with AR(1) dependence and phi = 0.75.

Usage

```
SimPois2_AR1_075_N1k
```

Format

A numeric vector of length 1000.

Source

Simulated for package testing.

sim_lgc *Simulate Count Time Series from Specified Model*

Description

Simulates a univariate count time series of length `n` from a specified model defined by a marginal distribution, ARMA dependence structure, and optional regression covariates.

Usage

```
sim_lgc(
  n,
  CountDist,
  MargParm,
  ARParm,
  MAParm,
  Regressor = NULL,
  Intercept = NULL,
  ntrials = NULL,
  ...
)
```

Arguments

`n`	Integer. Length of the time series to simulate.
`CountDist`	Character. Name of the marginal count distribution to simulate from. Supported values include `"Poisson"`, `"Negative Binomial"`, `"ZIP"`, `"Generalized Poisson"`, `"Mixed Poisson"`, `"Binomial"`, etc.
`MargParm`	Numeric vector. Parameters of the marginal distribution (e.g., mean, dispersion).

ARParm	Numeric vector. Autoregressive (AR) parameters. Can be empty or NULL for no AR component.
MAParm	Numeric vector. Moving average (MA) parameters. Can be empty or NULL for no MA component.
Regressor	Optional matrix or data frame of covariates. Used for dynamic regression models.
Intercept	Logical or numeric. Whether to include an intercept term. If numeric, should be 0 or 1.
ntrials	Integer. Number of trials for the Binomial model (required if CountDist == "Binomial").
...	Additional arguments (currently unused).

Details

The simulation is based on a latent Gaussian copula framework, where the marginal distribution is applied to transformed latent variables with dependence induced by an ARMA process.

Value

A ts object containing the simulated count time series.

Examples

```
# Simulate Poisson-ARMA(1,1) series
sim <- sim_lgc(
  n = 100,
  CountDist = "Poisson",
  MargParm = 3,
  ARParm = 0.5,
  MAParm = -0.3
)
ts.plot(sim)
```

---

summary.lgc *Summarize an lgc Model*

Description

Provides a summary of a fitted lgc model, including parameter estimates and fit statistics.

Usage

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

Arguments

object	An object of class lgc.
...	Additional arguments (currently unused).

Value

Invisibly returns the original `lgc` object.

Examples

```
## Not run:
  summary(mylgc)

## End(Not run)
```

Index