Package 'betaARMA'

May 24, 2026

Title Beta Autoregressive Moving Average Models

Version 1.2.0

Date 2026-04-22

Description Fits Beta Autoregressive Moving Average (BARMA) models for time series data distributed in the standard unit interval (0, 1). The estimation is performed via the conditional maximum likelihood method using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton algorithm. A ridge penalization scheme is available to improve numerical stability of the estimation, as proposed by Cribari-Neto, Costa and Fonseca (2025) <doi:10.1214/25-BJPS645>. The package includes tools for model fitting, diagnostic checking, and forecasting, along with two hydro-environmental datasets from Brazil. Based on the work of Rocha and Cribari-Neto (2009) <doi:10.1007/s11749-008-0112-z> and the associated erratum Rocha and Cribari-Neto (2017) <doi:10.1007/s11749-017-0528-4>. The original code was developed by Fabio M. Bayer.

License MIT + file LICENSE

URL https://github.com/Everton-da-Costa/betaARMA

BugReports https://github.com/Everton-da-Costa/betaARMA/issues

Encoding UTF-8

RoxygenNote 7.3.2

Language en-US

Imports forecast, ggplot2, rlang, gridExtra

Suggests knitr, zoo, xtable, here, moments, rmarkdown, tseries, lbfgs, dplyr, scales, testthat (>= 3.0.0)

VignetteBuilder knitr

Depends R (>= 3.5)

LazyData true

Config/testthat/edition 3

NeedsCompilation no

Author Everton da Costa [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7580-2639>),
Francisco Cribari-Neto [ctb, ths] (ORCID:
<https://orcid.org/0000-0002-5909-6698>, Theoretical foundations),
Vinicius Scher [ctb] (ORCID: <https://orcid.org/0000-0003-0406-0265>)

Maintainer Everton da Costa <everto.cost@gmail.com>

Repository CRAN

Date/Publication 2026-05-23 22:30:02 UTC

Contents

`barma` *Fit Beta Autoregressive Moving Average (BARMA) Models via Conditional Maximum Likelihood*

Description

Fits a Beta Autoregressive Moving Average (BARMA) model to time series data valued in (0, 1) using Conditional Maximum Likelihood Estimation (CMLE). The function performs complete model estimation including parameter estimation, hypothesis testing infrastructure, and model diagnostics.

Usage

```
barma(
  y,
  ar = integer(0),
  ma = integer(0),
  link = "logit",
  xreg = NULL,
  optimization = list(method = "BFGS", lower = -Inf, upper = Inf),
  penalty = FALSE
)
```

Arguments

`y`	A time series object (‘ts‘) with values strictly in the open interval (0, 1). Must have at least $\max(p, q) + 1$ observations.
`ar`	A numeric vector specifying autoregressive (AR) lags (e.g., ‘c(1, 2)‘ for AR(2)). Defaults to ‘integer(0)‘, which omits the AR component entirely. Absence should be expressed by omitting this argument or passing ‘integer(0)‘.
`ma`	A numeric vector specifying moving average (MA) lags (e.g., ‘1‘ for MA(1)). Defaults to ‘integer(0)‘, which omits the MA component entirely. Absence should be expressed by omitting this argument or passing ‘integer(0)‘.
`link`	The link function connecting the mean μ_t to the linear predictor η_t. One of: • ‘"logit"‘ (default): $g(x) = \log(x/(1-x))$ • ‘"probit"‘: $g(x) = \Phi^{-1}(x)$ • ‘"cloglog"‘: Complementary log-log link • ‘"loglog"‘: Log-log link
`xreg`	A matrix or data frame of external regressors (covariates), optional. Must have the same number of rows as ‘y‘. If provided, its columns are included in the linear predictor with associated coefficients in ‘beta‘.
`optimization`	A named list controlling the optimization procedure. Recognized fields: `method` Character string specifying the algorithm. One of `"BFGS"` (default), `"L-BFGS-B"`, or `"lbfgs"`. `lower` Numeric vector or scalar of lower bounds, used only by `"L-BFGS-B"`. Defaults to `-Inf`. `upper` Numeric vector or scalar of upper bounds, used only by `"L-BFGS-B"`. Defaults to `Inf`.
`penalty`	Logical. If `TRUE`, the ridge penalization scheme of Cribari-Neto, Costa and Fonseca (2025) is applied during conditional maximum likelihood estimation. The penalty term $(n-a)\lambda_n\|\boldsymbol{\nu}\|^2$ is subtracted from the conditional log-likelihood, where $\lambda_n = 1/(n-a)^{0.9}$ and $\boldsymbol{\nu}$ collects α, $\boldsymbol{\varphi}$, and $\boldsymbol{\theta}$. The precision parameter ϕ and regression coefficients $\boldsymbol{\beta}$ are excluded from penalization. Ridge penalization enhances the curvature of the conditional log-likelihood surface, reducing convergence failures and the occurrence of implausible estimates. Defaults to `FALSE`.

Details

Fit Beta Autoregressive Moving Average (BARMA) Models

This function fits the BARMA(p,q) model as proposed by Rocha & Cribari-Neto (2009, with erratum 2017). It serves as the main wrapper for the optimization process, calling specialized helper functions for likelihood computation, gradient calculation, and Fisher Information Matrix estimation.

Model Specification: The BARMA model is defined as:

$$g(\mu_t) = \alpha + X_t\beta + \sum_{i=1}^{p} \varphi_i(g(y_{t-i}) - X_{t-i}\beta) + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$$

where $y_t|F_{t-1} \sim Beta(\mu_t\phi, (1-\mu_t)\phi)$, $g(\cdot)$ is the link function, and F_{t-1} is the information set at time t-1.

Model Types (specified via ‘ar‘ and ‘ma‘ arguments):

- **BARMA(p,q):** Both ‘ar‘ and ‘ma‘ are specified.
- **BAR(p):** Only ‘ar‘ is specified; omit ‘ma‘ or pass ‘integer(0)‘.
- **BMA(q):** Only ‘ma‘ is specified; omit ‘ar‘ or pass ‘integer(0)‘.

External Regressors: Covariates can be included via ‘xreg‘. The model becomes a regression BARMA, where the mean depends on current covariates and lagged responses/errors.

Optimization: The function uses quasi-Newton algorithms (BFGS or L-BFGS-B) via `optim`, or the `lbfgs` function, utilizing analytic gradients for efficiency. Initial values are obtained from the internal ‘start_values()‘ function.

Implementation Notes: - The conditional log-likelihood is computed by conditioning on the first $m = \max(p, q)$ observations, which are required to initialize the recursive structure of the model. - The 2017 Erratum corrections are implemented for correct handling of moving average components in the score vector and Fisher Information Matrix. - All computations are vectorized where possible for efficiency.

Value

An object of class ‘"barma"‘ containing:

`coef`	Named vector of all estimated parameters, ordered as: alpha, AR parameters, MA parameters, beta parameters, phi.
`vcov`	The variance-covariance matrix of the estimators, computed as the inverse of the observed Fisher Information Matrix.
`model`	A summary table with coefficients, standard errors, z-statistics, and p-values for hypothesis tests $H_0 : \theta_i = 0$.
`fitted`	Fitted conditional mean values as a ‘ts‘ object (NA-padded for the first $m = \max(p, q)$ observations).
`muhat`	Alias for ‘fitted‘ (fitted mean values).
`etahat`	Estimated linear predictor values (full vector, NA-padded).
`errorhat`	Estimated errors on predictor scale (full vector, 0-padded).

`loglik`	The conditional log-likelihood at the CMLE.
`fisher_info_mat`	The expected Fisher Information Matrix (Rocha & Cribari-Neto, 2009).
`conv`	Convergence code from 'optim' (0 = success).
`alpha`, `beta`, `varphi`, `theta`, `phi`	Individual parameter estimates.
`start_values`	Initial parameter values used in optimization.
`call`	The original function call.
`opt`	Cleaned output list from the optimization procedure.
`opt_raw`	Raw output object returned directly by 'optim()' or 'lbfgs()'.

Note

The original version of this function was developed by Fabio M. Bayer (Federal University of Santa Maria, <bayer@ufsm.br>). It has been substantially modified and improved by Everton da Costa, with suggestions and contributions from Francisco Cribari-Neto.

Author(s)

Everton da Costa (Federal University of Pernambuco, <everto.cost@gmail.com>); Francisco Cribari-Neto (Federal University of Pernambuco, <francisco.cribari@ufpe.br>)

References

Rocha, A.V., & Cribari-Neto, F. (2009). Beta autoregressive moving average models. *TEST*, 18(3), 529-545. doi:10.1007/s117490080112z

Rocha, A.V., & Cribari-Neto, F. (2017). Erratum to: Beta autoregressive moving average models. *TEST*, 26, 451-459. doi:10.1007/s1174901705284

Cribari-Neto, F., Costa, E., & Fonseca, R.V. (2025). Numerical stability enhancements in beta autoregressive moving average model estimation. *Brazilian Journal of Probability and Statistics*, 39(4), 410-437. doi:10.1214/25BJPS645

See Also

`simu_barma` for simulation, `loglik_barma` for likelihood computation, `score_vector_barma` for gradient computation, `fim_barma` for Fisher Information Matrix

Examples

```
# Example 1: Fit a BAR(1) model (no MA component)
set.seed(2025)
y_sim_bar1 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  varphi = 0.6,
  phi    = 25.0,
  link   = "logit",
  freq   = 12
```

```
)

fit_bar1 <- barma(y_sim_bar1, ar = 1, link = "logit")
summary(fit_bar1)
coef(fit_bar1)

# Example 2: Fit a BARMA(1, 1) model
set.seed(2025)
y_sim_barma11 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  varphi = 0.6,
  theta  = 0.3,
  phi    = 25.0,
  link   = "logit",
  freq   = 12
)

fit_barma11 <- barma(y_sim_barma11, ar = 1, ma = 1, link = "logit")
summary(fit_barma11)

# Example 3: Fit a BMA(1) model (no AR component)
set.seed(2025)
y_sim_bma1 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  theta  = 0.3,
  phi    = 20.0,
  link   = "logit",
  freq   = 12
)

fit_bma1 <- barma(y_sim_bma1, ma = 1, link = "logit")
summary(fit_bma1)

# Example 4: BARMA(1, 1) model with harmonic seasonal regressors
hs <- sin(2 * pi * seq_along(y_sim_barma11) / 12)
hc <- cos(2 * pi * seq_along(y_sim_barma11) / 12)
X <- cbind(hs = hs, hc = hc)

fit_barma11_xreg <- barma(
  y_sim_barma11,
  ar   = 1,
  ma   = 1,
  link = "logit",
  xreg = X
)
summary(fit_barma11_xreg)

# Example 5: Fit a BARMA(1, 1) model using L-BFGS-B
set.seed(2025)
y_sim_barma11_LBFGSB <- simu_barma(
  n      = 250,
```

```
  alpha  = 0.0,
  varphi = 0.6,
  theta  = 0.3,
  phi    = 25.0,
  link   = "logit",
  freq   = 12
)

fit_barma11_LBFGSB <- barma(
  y_sim_barma11_LBFGSB,
  ar = 1,
  ma = 1,
  link = "logit",
  optimization = list(
    method = "L-BFGS-B",
    upper = c(Inf, 1, 1, Inf)
  )
)

summary(fit_barma11_LBFGSB)
```

`brasilia_df` *Monthly relative humidity in Brasília (data frame)*

Description

The same series as `brasilia_ts`, stored as a data frame with a `yearmon` time column.

Usage

```
brasilia_df
```

Format

A data frame with 2 columns:

time Month of observation (`yearmon`).

y Relative humidity as a proportion (numeric, in (0, 1)).

Source

NASA Prediction of Worldwide Energy Resources (POWER) project. https://power.larc.nasa.gov/

See Also

`brasilia_ts` for the same data as a `ts` object.

Examples

```
data(brasilia_df)
head(brasilia_df)
```

`brasilia_ts` *Monthly relative humidity in Brasília*

Description

Monthly relative humidity in Brasília, Brazil, expressed as a proportion. The data were obtained from the NASA POWER project and cover January 1999 to June 2024.

Usage

```
brasilia_ts
```

Format

A `ts` object with monthly observations, frequency 12, starting in January 1999.

Source

NASA Prediction of Worldwide Energy Resources (POWER) project. https://power.larc.nasa.gov/

References

Cribari-Neto, F., Costa, E., & Fonseca, R. V. (2025). Numerical stability enhancements in beta autoregressive moving average model estimation. *Brazilian Journal of Probability and Statistics*, 39(4), 410–437. doi:10.1214/25BJPS645

See Also

`brasilia_df` for the same data as a data frame.

Examples

```
data(brasilia_ts)
plot(brasilia_ts, ylab = "Relative humidity (proportion)", xlab = "Year")
```

`coef.barma` *Extract Coefficients from a barma Model*

Description

S3 method for extracting the vector of estimated coefficients from a fitted model object of class ‘"barma"‘.

Usage

```
## S3 method for class 'barma'
coef(object, ...)
```

Arguments

`object`	A fitted model object of class ‘"barma"‘.
`...`	Additional arguments (currently ignored).

Value

A named numeric vector of all estimated coefficients (e.g., alpha, varphi, theta, phi).

`fim_barma` *Compute Fisher Information Matrix for Beta Autoregressive Moving Average Models*

Description

Computes the expected Fisher Information Matrix (FIM) of a Beta Autoregressive Moving Average (BARMA) model. This function also efficiently returns auxiliary values like fitted values and residuals.

This function is designed for users who:

- Compute standard errors of parameter estimates
- Construct confidence intervals
- Perform hypothesis tests
- Verify theoretical properties
- Conduct simulation studies

Usage

```
fim_barma(
  y,
  ar = integer(0),
  ma = integer(0),
  alpha,
  varphi = numeric(0),
  theta = numeric(0),
  phi,
  link = "logit",
  xreg = NULL,
  beta = NULL,
  penalty = FALSE
)
```

Arguments

y	A numeric vector representing the time series data, in (0, 1).
ar	A numeric vector specifying the autoregressive (AR) lags (e.g., c(1, 2)). Defaults to integer(0), which omits the AR component entirely. Absence should be expressed by omitting this argument or passing integer(0).
ma	A numeric vector specifying the moving average (MA) lags (e.g., 1). Defaults to integer(0), which omits the MA component entirely. Absence should be expressed by omitting this argument or passing integer(0).
alpha	The intercept term (numeric scalar).
varphi	A numeric vector of autoregressive (AR) parameters. Absence should be expressed by omitting this argument or passing numeric(0).
theta	A numeric vector of moving average (MA) parameters. Absence should be expressed by omitting this argument or passing numeric(0).
phi	The precision parameter of the BARMA model (must be positive and finite). Larger values indicate less variance for a given mean.
link	A character string specifying the link function: "logit" (default), "probit", or "cloglog".
xreg	A matrix or data frame of static regressors (optional). Must have the same number of rows as length of y.
beta	A numeric vector of regression coefficients for xreg (optional). Length must match number of columns in xreg.
penalty	Logical. If TRUE, the ridge penalty of Cribari-Neto, Costa and Fonseca (2025) is added to the Fisher Information Matrix, yielding the penalized FIM $K_{\text{pen}}(\boldsymbol{\nu}) = K(\boldsymbol{\nu}) + 2(n-a)\Lambda$, where $\Lambda = \text{diag}\{\lambda_n, \ldots, \lambda_n, 0, \ldots, 0\}$ with $\lambda_n = 1/(n-a)^{0.9}$ applied to α, $\boldsymbol{\varphi}$, and $\boldsymbol{\theta}$ only. Defaults to FALSE.

Details

Fisher Information Matrix for BARMA Models

The expected Fisher Information Matrix is computed analytically using the trigamma-based expressions from Rocha and Cribari-Neto (2009). Standard errors are obtained via `sqrt(diag(solve(FIM)))`.

Non-Diagonal Structure: Unlike generalized linear models, the FIM is not block-diagonal due to the coupling introduced by the ARMA dynamics. This is documented in the Rocha & Cribari-Neto (2009) paper.

Important: This function implements the corrections from the 2017 Erratum (Rocha & Cribari-Neto, 2017) for moving average components. See References section for details.

Parameter Order: Parameters should be supplied in the order: alpha, varphi (AR), theta (MA), beta (regressors), phi. This matches the parameter order used by `barma`.

Value

A list containing:

`fisher_info_mat`
: The Fisher Information Matrix (numeric matrix). Dimensions: `(k+p+q+2)` x `(k+p+q+2)` where k is number of regressors, p is AR order, q is MA order. The matrix is symmetric and should be positive definite at the MLE.

`fitted_ts`
: The fitted values (conditional mean) as a `ts` object, with the same time index as the input `y`. Values for the first $m = \max(p, q)$ observations are `NA`.

`muhat_effective`
: The fitted conditional means (numeric vector) excluding the first m observations.

`etahat_full`
: The estimated linear predictor values (numeric vector, length = length(y)), with `NA` for the first m observations.

`errorhat_full`
: The estimated errors on the predictor scale (numeric vector, length = length(y)), zero-padded for the first m observations.

References

Rocha, A.V., & Cribari-Neto, F. (2009). Beta autoregressive moving average models. *TEST*, 18(3), 529-545. doi:10.1007/s117490080112z

Rocha, A.V., & Cribari-Neto, F. (2017). Erratum to: Beta autoregressive moving average models. *TEST*, 26, 451-459. doi:10.1007/s1174901705284

Cribari-Neto, F., Costa, E., & Fonseca, R.V. (2025). Numerical stability enhancements in beta autoregressive moving average model estimation. *Brazilian Journal of Probability and Statistics*, 39(4), 410-437. doi:10.1214/25BJPS645

See Also

`barma` for model fitting, `loglik_barma` for log-likelihood computation, `score_vector_barma` for score vector (gradient)

Examples

```
# Example 1: Fisher Information Matrix for a BAR(1) model (no MA component)
set.seed(2025)
y_sim_bar1 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  varphi = 0.6,
  phi    = 25.0,
  link   = "logit",
  freq   = 12
)

result_bar1 <- fim_barma(
  y      = y_sim_bar1,
  ar     = 1,
  alpha  = 0.0,
  varphi = 0.6,
  theta  = numeric(0),
  phi    = 25.0,
  link   = "logit"
)

# Check positive definiteness
fim <- result_bar1$fisher_info_mat
all(eigen(fim)$values > 0)

# Standard errors from inverse of FIM
sqrt(diag(solve(fim)))

# Example 2: Fisher Information Matrix for a BARMA(1, 1) model
set.seed(2025)
y_sim_barma11 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  varphi = 0.6,
  theta  = 0.3,
  phi    = 25.0,
  link   = "logit",
  freq   = 12
)

result_barma11 <- fim_barma(
  y      = y_sim_barma11,
  ar     = 1,
  ma     = 1,
  alpha  = 0.0,
  varphi = 0.6,
  theta  = 0.3,
  phi    = 25.0,
  link   = "logit"
)
```

```
sqrt(diag(solve(result_barma11$fisher_info_mat)))

# Example 3: Fisher Information Matrix for a BMA(1) model (no AR component)
set.seed(2025)
y_sim_bma1 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  theta  = 0.3,
  phi    = 20.0,
  link   = "logit",
  freq   = 12
)

result_bma1 <- fim_barma(
  y      = y_sim_bma1,
  ma     = 1,
  alpha  = 0.0,
  varphi = numeric(0),
  theta  = 0.3,
  phi    = 20.0,
  link   = "logit"
)

sqrt(diag(solve(result_bma1$fisher_info_mat)))
```

`fitted.barma` *Extract Fitted Values from a barma Model*

Description

S3 method for extracting the fitted mean values (mu-hat) from a fitted model object of class ‘"barma"‘.

Usage

```
## S3 method for class 'barma'
fitted(object, ...)
```

Arguments

`object`	A fitted model object of class ‘"barma"‘.
`...`	Additional arguments (currently ignored).

Details

The fitted values are returned as a time series (‘ts‘) object, matching the time properties of the original input ‘y‘. The first ‘max_lag‘ observations are ‘NA‘, as they cannot be fitted.

Value

A ‘ts‘ object of the fitted mean values.

`forecast.barma` *Forecast a barma Model*

Description

S3 method for producing forecasts from a fitted ‘barma‘ model object.

Usage

```
## S3 method for class 'barma'
forecast(object, h = 6, xreg = NULL, ...)
```

Arguments

`object`	A fitted model object of class ‘barma‘. Must contain ‘object$xreg‘ if regressors were used.
`h`	The number of steps to forecast ahead (forecast horizon). Default is 6.
`xreg`	A matrix of future regressor values for the forecast horizon. Should have h rows and the same number of columns as xreg used in model fitting.
`...`	Additional arguments (currently ignored).

Details

This function computes dynamic, multi-step-ahead point forecasts. It implements a "Regression with ARMA errors" logic, where the AR components are applied to the deviations from the regression line: $AR(g(y_{t-k}) - x_{t-k}^{\top}\beta)$.

Value

A ‘ts‘ object containing the point forecasts for h steps ahead.

`loglik_barma` *Compute Conditional Log-Likelihood for Beta Autoregressive Moving Average Models*

Description

Computes the conditional log-likelihood of a Beta Autoregressive Moving Average (BARMA) model. This function is designed for users who:

- Implement custom optimization algorithms
- Verify theoretical properties
- Conduct simulation studies
- Debug model fitting
- Integrate BARMA models into their own workflows

Usage

```
loglik_barma(
  y,
  ar = integer(0),
  ma = integer(0),
  alpha,
  varphi = numeric(0),
  theta = numeric(0),
  phi,
  link = "logit",
  xreg = NULL,
  beta = NULL,
  penalty = FALSE
)
```

Arguments

`y`	A numeric vector representing the time series data, with values strictly in (0, 1).
`ar`	A numeric vector specifying the autoregressive (AR) lags (e.g., `c(1, 2)`). Defaults to `integer(0)`, which omits the AR component entirely. Absence should be expressed by omitting this argument or passing `integer(0)`.
`ma`	A numeric vector specifying the moving average (MA) lags (e.g., `1`). Defaults to `integer(0)`, which omits the MA component entirely. Absence should be expressed by omitting this argument or passing `integer(0)`.
`alpha`	The intercept term (numeric scalar).
`varphi`	A numeric vector of autoregressive (AR) parameters. Absence should be expressed by omitting this argument or passing `numeric(0)`.
`theta`	A numeric vector of moving average (MA) parameters. Absence should be expressed by omitting this argument or passing `numeric(0)`.

phi	The precision parameter of the BARMA model (must be positive and finite). Larger values indicate less variance for a given mean.
link	A character string specifying the link function: "logit" (default), "probit", or "cloglog".
xreg	A matrix or data frame of static regressors (optional). Must have the same number of rows as length of y.
beta	A numeric vector of regression coefficients for xreg (optional). Length must match number of columns in xreg.
penalty	Logical. If TRUE, a ridge penalty is subtracted from the conditional log-likelihood, yielding the penalized conditional log-likelihood proposed by Cribari-Neto, Costa and Fonseca (2025). The penalty term is $(n-a)\lambda_n\|\boldsymbol{\nu}\|^2$, where $\lambda_n = 1/(n-a)^{0.9}$ and $\boldsymbol{\nu}$ collects α, $\boldsymbol{\varphi}$, and $\boldsymbol{\theta}$ (the precision parameter ϕ and regression coefficients $\boldsymbol{\beta}$ are excluded). Defaults to FALSE.

Details

Log-Likelihood for BARMA Models

The log-likelihood is computed as:

$$\ell = \sum_{t=m+1}^{n} \log f(y_t|\mu_t, \phi)$$

where f is the Beta density with shape parameters $shape1 = \mu_t * \phi$ and $shape2 = (1-\mu_t) * \phi$.

The linear predictor is constructed as:

$$\eta_t = \alpha + X_t\beta + \sum_{i=1}^{p} \varphi_i(y_{t-i} - X_{t-i}\beta) + \sum_{j=1}^{q} \theta_j\epsilon_{t-j}$$

Important: This function implements the corrections from the 2017 Erratum (Rocha & Cribari-Neto, 2017) for moving average components. See References section for details.

Parameter Order: Parameters should be supplied in the order: alpha, varphi (AR), theta (MA), phi, beta (regressors). This matches the parameter order used by barma.

Value

A numeric scalar representing the (penalized) conditional log-likelihood value. When penalty = FALSE (default), the standard conditional log-likelihood is returned. When penalty = TRUE, the ridge-penalized conditional log-likelihood of Cribari-Neto, Costa and Fonseca (2025) is returned instead. Returns -Inf if:

- phi is non-positive or non-finite
- Insufficient observations for specified lag structure
- Fitted values are outside (0, 1)
- Any numerical issues occur

References

Rocha, A.V., & Cribari-Neto, F. (2009). Beta autoregressive moving average models. *TEST*, 18(3), 529-545. doi:10.1007/s117490080112z

Rocha, A.V., & Cribari-Neto, F. (2017). Erratum to: Beta autoregressive moving average models. *TEST*, 26, 451-459. doi:10.1007/s1174901705284

Cribari-Neto, F., Costa, E., & Fonseca, R.V. (2025). Numerical stability enhancements in beta autoregressive moving average model estimation. *Brazilian Journal of Probability and Statistics*, 39(4), 410-437. doi:10.1214/25BJPS645

See Also

`barma` for model fitting, `score_vector_barma` for gradient computation, `fim_barma` for Fisher Information Matrix

Examples

```
# Example 1: Log-likelihood for a BAR(1) model (no MA component)
set.seed(2025)
y_sim_bar1 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  varphi = 0.6,
  phi    = 25.0,
  link   = "logit",
  freq   = 12
)

loglik_barma(
  y      = y_sim_bar1,
  ar     = 1,
  alpha  = 0.0,
  varphi = 0.6,
  theta  = numeric(0),
  phi    = 25.0,
  link   = "logit"
)

# Example 2: Log-likelihood for a BARMA(1, 1) model
set.seed(2025)
y_sim_barma11 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  varphi = 0.6,
  theta  = 0.3,
  phi    = 25.0,
  link   = "logit",
  freq   = 12
)

loglik_barma(
  y      = y_sim_barma11,
```

```
  ar     = 1,
  ma     = 1,
  alpha  = 0.0,
  varphi = 0.6,
  theta  = 0.3,
  phi    = 25.0,
  link   = "logit"
)

# Example 3: Log-likelihood for a BMA(1) model (no AR component)
set.seed(2025)
y_sim_bma1 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  theta  = 0.3,
  phi    = 20.0,
  link   = "logit",
  freq   = 12
)

loglik_barma(
  y      = y_sim_bma1,
  ma     = 1,
  alpha  = 0.0,
  varphi = numeric(0),
  theta  = 0.3,
  phi    = 20.0,
  link   = "logit"
)

# Example 4: Ridge-penalized log-likelihood for a BARMA(1, 1) model
# Uses penalty = TRUE as recommended by Cribari-Neto, Costa and
# Fonseca (2025, BJPS) to improve numerical stability.
loglik_barma(
  y       = y_sim_barma11,
  ar      = 1,
  ma      = 1,
  alpha   = 0.0,
  varphi  = 0.6,
  theta   = 0.3,
  phi     = 25.0,
  link    = "logit",
  penalty = TRUE
)
```

---

make_link_structure *Create Link Function Structure for BARMA Models*

Description

A helper function that constructs a list containing the link function, its inverse, and the derivative of the mean function. It extends the standard `make.link` by adding support for the `"loglog"` link.

Usage

```
make_link_structure(link = "logit")
```

Arguments

`link`	A character string specifying the link function. Defaults to `"logit"`. Accepted values are `"logit"`, `"probit"`, `"cloglog"`, and `"loglog"`.

Details

This function is used by `barma`, `loglik_barma`, `score_vector_barma`, and `fim_barma` to handle the `link` argument in a standardized way. It is also exported for users who implement custom workflows.

For the `"logit"`, `"probit"`, and `"cloglog"` links, the function acts as a wrapper around the base R `make.link`.

For the `"loglog"` link, which is not available in `make.link`, the necessary components are defined explicitly:

- Link function: $g(\mu) = -\log(-\log(\mu))$
- Inverse link: $g^{-1}(\eta) = \exp(-\exp(-\eta))$
- Derivative $\frac{d\mu}{d\eta}$: $\exp(-\exp(-\eta) - \eta)$

If an unsupported link is provided, the function stops with an error message listing the available options.

Value

A list with three components:

`linkfun`	The link function $g(\mu)$, transforming $\mu \in (0, 1)$ to $\eta \in (-\infty, \infty)$.
`linkinv`	The inverse link function $g^{-1}(\eta)$, transforming η back to μ.
`mu.eta`	The derivative $d\mu/d\eta$ of the inverse link function.

Author(s)

Original R code by Fabio M. Bayer (Federal University of Santa Maria, <bayer@ufsm.br>). Modified and improved by Everton da Costa (Federal University of Pernambuco, <everto.cost@gmail.com>).

See Also

`barma`, `make.link`

Examples

```
# --- Create a logit link structure ---
logit_link <- make_link_structure(link = "logit")

# Apply the link function
mu <- 0.5
eta <- logit_link$linkfun(mu)
print(eta) # Should be 0

# Apply the inverse link function
mu_restored <- logit_link$linkinv(eta)
print(mu_restored) # Should be 0.5

# --- Create a loglog link structure ---
loglog_link <- make_link_structure(link = "loglog")

# Apply the loglog link function
mu_loglog <- 0.8
eta_loglog <- loglog_link$linkfun(mu_loglog)
print(eta_loglog)

# Apply the inverse loglog link function
mu_restored_loglog <- loglog_link$linkinv(eta_loglog)
print(mu_restored_loglog) # Should be ~0.8
```

`plot.barma` *Diagnostic Plots for a Fitted betaARMA Model*

Description

Produces diagnostic plots for a βARMA model fitted by `barma`. By default (`which = "default"`), four panels are displayed in a 2×2 grid: observed vs. fitted values, residual ACF, Ljung-Box p-values, and residual PACF.

Usage

```
## S3 method for class 'barma'
plot(
  x,
  which = c("default", "all", "fitted", "tsplot", "acf", "pacf", "ljungbox", "monti",
    "hist", "qq"),
  residual_type = c("pearson", "quantile", "link", "raw"),
  lag_max = 24,
  title = NULL,
  colour_observed = "#00BFC4",
  colour_fitted = "#F8766D",
  colour_residual = "#7CAE00",
  ...
)
```

Arguments

x	An object of class "barma", as returned by barma.
which	A character string controlling which panel(s) to display. One of:
	"default" **(default)** Four-panel 2×2 grid: observed vs. fitted (top left), residual ACF (top right), Ljung-Box p-values (bottom left), and residual PACF (bottom right).
	"all" Six-panel 3×2 grid: observed vs. fitted (top left), residuals over time (top right), residual ACF (middle left), residual PACF (middle right), Ljung-Box p-values (bottom left), Monti p-values (bottom right).
	"fitted" Single panel: observed vs. fitted values.
	"tsplot" Single panel: residuals plotted as a time series, with a horizontal reference line at zero and dashed lines at ±3 (ad hoc threshold; not shown for "raw" residuals).
	"acf" Single panel: residual ACF.
	"pacf" Single panel: residual PACF.
	"ljungbox" Single panel: Ljung-Box p-values.
	"monti" Single panel: Monti test p-values.
	"hist" Single panel: residual distribution histogram with kernel density overlay.
	"qq" Single panel: Normal Q-Q plot. Always uses quantile residuals regardless of residual_type; a message is issued if the type is overridden.
residual_type	Character string passed to residuals.barma controlling which residuals are computed and displayed. One of "pearson" (default), "quantile", "link" or "raw". Pearson residuals are recommended for ACF/PACF analysis and portmanteau tests (Scher, Cribari-Neto, Pumi, and Bayer, 2020); quantile residuals are recommended for normality assessment (Dunn and Smyth, 1996). See also Ferrari and Cribari-Neto (2004).
lag_max	Maximum lag to display in the ACF, PACF, Ljung-Box, and Monti panels. Defaults to 24, appropriate for monthly series.
title	An optional character string used as the overall title of the plot. Defaults to NULL (no title).
colour_observed	Colour for observed values in the fitted-values panel. Defaults to "#00BFC4" (teal).
colour_fitted	Colour for fitted values in the fitted-values panel. Defaults to "#F8766D" (red).
colour_residual	Colour for residual lines, ACF/PACF bars, portmanteau test points, and histogram fill. Defaults to "#7CAE00" (green).
...	Additional arguments (currently unused, included for S3 consistency).

Details

Residual type. All panels (except *Observed vs. Fitted*) use the residuals selected by residual_type. The residual type is displayed in each panel subtitle for transparency.

Reference lines in the residuals-over-time panel. The dashed horizontal lines are drawn at ± 3 as an ad hoc threshold for identifying atypical observations. No distributional assumption is made; observations outside these bounds may warrant individual inspection. Pearson and link-scale residuals do not follow a standard normal distribution, so normal-distribution-based quantiles (e.g., ± 1.96 or ± 2.576) are not appropriate reference values for such residuals. Quantile residuals are approximately distributed as $N(0, 1)$ under a correctly specified model, so ± 3 is a conservative but reasonable threshold for them.

Q-Q plot residuals. The `"qq"` panel always uses quantile residuals (Dunn and Smyth, 1996), regardless of `residual_type`. Quantile residuals are the only type theoretically expected to follow $N(0, 1)$ under a correctly specified model; using Pearson, raw, or link-scale residuals in a normal Q-Q plot would produce misleading results. A message is issued when `residual_type` is overridden.

Effective sample size. Both the Ljung-Box and Monti test statistics rely on the effective sample size $N_{eff} = n - \max(p, q)$ (where $\max(p, q)$ corresponds to the maximum lag in the model).

Ljung-Box test. The p-values are computed from the Ljung-Box test statistic. $Q_{LB}(k) = N_{eff}(N_{eff} + 2) \sum_{j=1}^{k} \hat{\rho}_j^2/(N_{eff} - j)$, where $\hat{\rho}_j$ is the j-th sample autocorrelation of the residuals and $N_{eff} = n - \max(p, q)$ is the effective sample size. The degrees of freedom are adjusted by subtracting the total number of estimated AR and MA parameters, $n_{ar} + n_{ma}$ (Scher, Cribari-Neto, Pumi, and Bayer (2020)). The first $n_{ar} + n_{ma}$ lags are set to `NA` since the test statistic is not defined for $k \leq n_{ar} + n_{ma}$.

Monti test. The p-values are computed from the test statistic $M(k) = N_{eff}(N_{eff}+2) \sum_{j=1}^{k} \hat{\rho}_{jj}^2/(N_{eff}-j)$, where $\hat{\rho}_{jj}$ is the j-th sample partial autocorrelation of the residuals, and $N_{eff} = n - \max\_lag$ is the effective sample size after stripping initial NAs. The degrees of freedom are adjusted to $k - (n_{ar} + n_{ma})$ (Monti, 1994; Scher, Cribari-Neto, Pumi, and Bayer (2020)). The first $n_{ar} + n_{ma}$ lags are set to `NA` as the statistic is not defined for $k \leq n_{ar} + n_{ma}$.

Value

For grid requests (`"default"` or `"all"`), invisibly returns a `gtable` object produced by `gridExtra::arrangeGrob`, which can be saved with `ggplot2::ggsave`. For single-panel requests, invisibly returns the individual `ggplot` object.

References

Dunn, P. K., & Smyth, G. K. (1996). Randomized quantile residuals. *Journal of Computational and Graphical Statistics*, 5(3), 236–244. doi:10.1080/10618600.1996.10474708

Ferrari, S. L. P., & Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions. *Journal of Applied Statistics*, 31(7), 799–815. doi:10.1080/0266476042000214501

Monti, A. C. (1994). A proposal for a residual autocorrelation test in linear models. *Biometrika*, 81(4), 776–780. doi:10.1093/biomet/81.4.776

Rocha, A. V., & Cribari-Neto, F. (2009). Beta autoregressive moving average models. *TEST*, 18(3), 529–545. doi:10.1007/s117490080112z

Rocha, A. V., & Cribari-Neto, F. (2017). Erratum to: Beta autoregressive moving average models. *TEST*, 26, 451–459. doi:10.1007/s1174901705284

See Also

barma for model fitting; residuals.barma for the residual extraction method; fitted.barma for the fitted-value extraction method; forecast.barma for out-of-sample forecasting.

Examples

```
data(brasilia_ts)
fit <- barma(brasilia_ts, ar = 1, ma = 1)

# Default four-panel diagnostic grid (Pearson residuals)
plot(fit)

# Six-panel diagnostic grid
plot(fit, which = "all")

# Single panel: residuals as time series
plot(fit, which = "tsplot")

# Single panel: Ljung-Box p-values only
plot(fit, which = "ljungbox")

# Single panel: Monti p-values only
plot(fit, which = "monti")

# Normal Q-Q plot (always uses quantile residuals)
plot(fit, which = "qq")

# Histogram with quantile residuals
plot(fit, which = "hist", residual_type = "quantile")
```

---

print.barma *Print Method for a barma Model*

Description

S3 method for printing a summary of a fitted ‘"barma"‘ model object.

Usage

```
## S3 method for class 'barma'
print(x, ...)
```

Arguments

x	A fitted model object of class ‘"barma"‘.
...	Additional arguments (currently ignored).

Details

This function provides a concise summary, showing the function call that generated the model, the link function used, and the final estimated coefficients.

Value

Invisibly returns the original object ‘x‘.

`print.summary.barma` *Print Method for a barma Summary*

Description

S3 method for printing the detailed summary of a ‘"barma"‘ model object.

Usage

```
## S3 method for class 'summary.barma'
print(x, ...)
```

Arguments

`x`	A fitted model summary object of class ‘"summary.barma"‘.
`...`	Additional arguments (currently ignored).

Details

This function formats and displays the summary list created by ‘summary.barma()‘, including the call, coefficient table, and information criteria.

Value

Invisibly returns the original object ‘x‘.

`residuals.barma` *Residuals for a Fitted betaARMA Model*

Description

Computes residuals for a fitted βARMA model object of class `"barma"`. Four types are available, controlled by the `type` argument.

Usage

```
## S3 method for class 'barma'
residuals(object, type = c("pearson", "quantile", "link", "raw"), ...)
```

Arguments

`object`	A fitted model object of class `"barma"`, as returned by `barma`.
`type`	Character string specifying the type of residuals to compute. One of:
	`"pearson"` **(default)** Pearson residuals on the response scale, as defined in Ferrari and Cribari-Neto (2004). Recommended for ACF/PACF analysis and portmanteau tests.
	`"quantile"` Quantile residuals (Dunn and Smyth, 1996), approximately i.i.d. $N(0,1)$ under a correctly specified model. Recommended for normality assessment.
	`"link"` Residuals on the predictor (link) scale. Retained for compatibility with earlier versions of the package.
	`"raw"` Raw residuals $y_t - \hat{\mu}_t$ on the response scale.
`...`	Additional arguments (currently unused, included for S3 consistency).

Details

Pearson residuals (default) are defined on the response scale as:

$$r_t = \frac{y_t - \hat{\mu}_t}{\sqrt{\hat{\mu}_t(1-\hat{\mu}_t)/(1+\hat{\phi})}}$$

where $\hat{\mu}_t$ is the fitted conditional mean and $\hat{\phi}$ is the estimated precision parameter. This residual was defined in Ferrari and Cribari-Neto (2004) for beta regression models and is adopted here in the βARMA setting. These residuals are appropriate for ACF/PACF analysis and portmanteau tests (Ljung-Box and Monti).

Quantile residuals (Dunn and Smyth, 1996) are defined as:

$$r_t^Q = \Phi^{-1}\Big(F(y_t;\, \hat{\mu}_t\hat{\phi},\, (1-\hat{\mu}_t)\hat{\phi})\Big)$$

where $F(\cdot\,;\, a, b)$ is the Beta CDF with shape parameters $a = \hat{\mu}_t\hat{\phi}$ and $b = (1-\hat{\mu}_t)\hat{\phi}$, and Φ^{-1} is the standard normal quantile function. Under a correctly specified model, quantile residuals are

approximately i.i.d. $N(0,1)$, making them suitable for normality assessment via histograms and Q-Q plots.

Link-scale residuals are defined on the predictor scale as:

$$r_t^L = \frac{g(y_t) - \hat{\eta}_t}{\sqrt{[g'(\hat{\mu}_t)]^2 \cdot \hat{\mu}_t(1-\hat{\mu}_t)/(1+\hat{\phi})}}$$

where $g(\cdot)$ is the link function, $\hat{\eta}_t$ is the fitted linear predictor, and $g'(\hat{\mu}_t) = d\eta/d\mu$. This residual was used in earlier versions of the package and is retained for compatibility.

Raw residuals are the simple difference on the response scale:

$$r_t^{\text{raw}} = y_t - \hat{\mu}_t$$

Value

A numeric `ts` object of the same length as the response series, with the first `max_lag` values set to `NA`. The `ts` attributes (`start` and `frequency`) are inherited from the original response series stored in `object$y`.

References

Dunn, P. K., & Smyth, G. K. (1996). Randomized quantile residuals. *Journal of Computational and Graphical Statistics*, 5(3), 236–244. doi:10.1080/10618600.1996.10474708

Ferrari, S. L. P., & Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions. *Journal of Applied Statistics*, 31(7), 799–815. doi:10.1080/0266476042000214501

Rocha, A. V., & Cribari-Neto, F. (2009). Beta autoregressive moving average models. *TEST*, 18(3), 529–545. doi:10.1007/s117490080112z

Rocha, A. V., & Cribari-Neto, F. (2017). Erratum to: Beta autoregressive moving average models. *TEST*, 26, 451–459. doi:10.1007/s1174901705284

See Also

`barma` for model fitting; `fitted.barma` for fitted values; `plot.barma` for diagnostic plots.

Examples

```
data(brasilia_ts)
fit <- barma(brasilia_ts, ar = 1, ma = 1)

# Pearson residuals (default) -- Ferrari & Cribari-Neto (2004)
res <- residuals(fit)

# Raw residuals
res_raw <- residuals(fit, type = "raw")

# Quantile residuals -- recommended for normality assessment
res_q <- residuals(fit, type = "quantile")

# Link-scale residuals
```

```
res_link <- residuals(fit, type = "link")
```

`score_vector_barma` *Score Vector for the BARMA Model*

Description

Computes the score vector (gradient of the log-likelihood) for the Beta Autoregressive Moving Average (BARMA) model at a given parameter vector. This function is designed for users who:

- Implement custom optimization algorithms
- Verify theoretical properties
- Conduct simulation studies
- Debug model fitting
- Integrate BARMA models into their own workflows

Usage

```
score_vector_barma(
  y,
  ar = integer(0),
  ma = integer(0),
  alpha,
  varphi = numeric(0),
  theta = numeric(0),
  phi,
  link = "logit",
  xreg = NULL,
  beta = NULL,
  penalty = FALSE
)
```

Arguments

`y`	A numeric vector representing the time series data, with values strictly in (0, 1).
`ar`	A numeric vector specifying the autoregressive (AR) lags (e.g., `c(1, 2)`). Defaults to `integer(0)`, which omits the AR component entirely. Absence should be expressed by omitting this argument or passing `integer(0)`.
`ma`	A numeric vector specifying the moving average (MA) lags (e.g., `1`). Defaults to `integer(0)`, which omits the MA component entirely. Absence should be expressed by omitting this argument or passing `integer(0)`.
`alpha`	The intercept term (numeric scalar).
`varphi`	A numeric vector of autoregressive (AR) parameters. Absence should be expressed by omitting this argument or passing `numeric(0)`.

theta	A numeric vector of moving average (MA) parameters. Absence should be expressed by omitting this argument or passing `numeric(0)`.
phi	The precision parameter of the BARMA model (must be positive and finite). Larger values indicate less variance for a given mean.
link	A character string specifying the link function: `"logit"` (default), `"probit"`, or `"cloglog"`.
xreg	A matrix or data frame of static regressors (optional). Must have the same number of rows as length of `y`.
beta	A numeric vector of regression coefficients for `xreg` (optional). Length must match number of columns in `xreg`.
penalty	Logical. If `TRUE`, the gradient of the ridge penalty of Cribari-Neto, Costa and Fonseca (2025) is subtracted from the score vector, yielding the penalized score $S_{\text{pen}}(\boldsymbol{\nu}) = S(\boldsymbol{\nu}) - 2(n-a)\lambda_n\boldsymbol{\nu}$, where $\lambda_n = 1/(n-a)^{0.9}$ and $\boldsymbol{\nu}$ collects α, $\boldsymbol{\varphi}$, and $\boldsymbol{\theta}$ (ϕ and $\boldsymbol{\beta}$ are excluded). Defaults to `FALSE`.

Value

A numeric vector of the same length as the parameter vector, giving the partial derivatives of the log-likelihood with respect to each parameter. The order of the components is: `(alpha, varphi, theta, beta, phi)`.

See Also

barma, loglik_barma, fim_barma

Examples

```
# Example 1: Score vector for a BAR(1) model (no MA component)
set.seed(2025)
y_sim_bar1 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  varphi = 0.6,
  phi    = 25.0,
  link   = "logit",
  freq   = 12
)

score_vector_barma(
  y      = y_sim_bar1,
  ar     = 1,
  alpha  = 0.0,
  varphi = 0.6,
  theta  = numeric(0),
  phi    = 25.0,
  link   = "logit"
)

# Example 2: Score vector for a BARMA(1, 1) model
set.seed(2025)
```

```
y_sim_barma11 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  varphi = 0.6,
  theta  = 0.3,
  phi    = 25.0,
  link   = "logit",
  freq   = 12
)

score_vector_barma(
  y      = y_sim_barma11,
  ar     = 1,
  ma     = 1,
  alpha  = 0.0,
  varphi = 0.6,
  theta  = 0.3,
  phi    = 25.0,
  link   = "logit"
)

# Example 3: Score vector for a BMA(1) model (no AR component)
set.seed(2025)
y_sim_bma1 <- simu_barma(
  n      = 250,
  alpha  = 0.0,
  theta  = 0.3,
  phi    = 20.0,
  link   = "logit",
  freq   = 12
)

score_vector_barma(
  y      = y_sim_bma1,
  ma     = 1,
  alpha  = 0.0,
  varphi = numeric(0),
  theta  = 0.3,
  phi    = 20.0,
  link   = "logit"
)
```

---

simu_barma *Simulate a Beta Autoregressive Moving Average (BARMA) Time Series*

Description

Generates a random time series from a Beta Autoregressive Moving Average (BARMA) model. The function can simulate BARMA(p,q), BAR(p), or BMA(q) processes.

Usage

```
simu_barma(
  n,
  alpha = 0,
  varphi = NA,
  theta = NA,
  phi = 20,
  freq = 12,
  link = "logit"
)
```

Arguments

n	The desired length of the final time series (after burn-in).
alpha	The intercept term (α) in the linear predictor. Default is '0.0'.
varphi	A numeric vector of autoregressive (AR) parameters (φ). Default is 'NA' for models without an AR component.
theta	A numeric vector of moving average (MA) parameters (θ). Default is 'NA' for models without an MA component.
phi	The precision parameter ($\phi > 0$) of the Beta distribution. Higher values result in less variance. Default is '20'.
freq	The frequency of the resulting 'ts' object (e.g., 12 for monthly, 4 for quarterly). Default is '12'.
link	The link function to connect the mean μ_t to the linear predictor. Must be one of '"logit"' (default), '"probit"', '"cloglog"', or '"loglog"'.

Details

The model type is determined by the 'varphi' and 'theta' parameters:

- **BARMA(p,q):** Both 'varphi' and 'theta' are provided.
- **BAR(p):** Only 'varphi' is provided ('theta' is 'NA').
- **BMA(q):** Only 'theta' is provided ('varphi' is 'NA').

Value

A 'ts' object of length 'n' containing the simulated Beta-distributed time series.

Author(s)

Original R code by: Fabio M. Bayer (Federal University of Santa Maria, <bayer@ufsm.br>) Modified and improved by: Everton da Costa (Federal University of Pernambuco, <everto.cost@gmail.com>)

See Also

barma, make_link_structure

Examples

```
# Set seed for reproducibility
# --- Example 1: Simulate a BAR(1) process ---
# y_t depends on y_{t-1}
set.seed(2025)
bar1_series <- simu_barma(n = 250, alpha = 0.0, varphi = 0.5, phi = 20,
link = "logit")
plot(bar1_series, main = "Simulated BAR(1) Process", ylab = "Value")

# --- Example 2: Simulate a BMA(1) process ---
# y_t depends on the previous error term
set.seed(2025)
bma1_series <- simu_barma(n = 250, alpha = 0.0, theta = -0.2, phi = 20)
plot(bma1_series, main = "Simulated BMA(1) Process", ylab = "Value")

# --- Example 3: Simulate a BARMA(2,1) process with a cloglog link ---
set.seed(2025)
barma21_series <- simu_barma(
  n = 200,
  alpha = 0.0,
  varphi = c(0.4, 0.2), # AR(2) components
  theta = -0.3,         # MA(1) component
  phi = 20,
  link = "cloglog"
)
plot(barma21_series, main = "Simulated BARMA(2,1) Process", ylab = "Value")
```

`start_values` *Generate Initial Values for BARMA Model Estimation*

Description

This function calculates reasonable starting values for the parameters of various Beta Autoregressive Moving Average (BARMA) models. The method is based on the approach proposed by Ferrari & Cribari-Neto (2004) for beta regression, adapted here for the time series context.

Usage

```
start_values(y, link, ar = integer(0), ma = integer(0), xreg = NA)
```

Arguments

`y`	A numeric time series with values in the open interval (0, 1).
`link`	A string specifying the link function for the mean.
`ar`	A numeric vector of autoregressive (AR) lags. Defaults to `integer(0)`, which omits the AR component entirely.

`ma`	A numeric vector of moving average (MA) lags. Defaults to `integer(0)`, which omits the MA component entirely.
`xreg`	An optional numeric matrix or data frame of exogenous variables.

Details

The function computes initial values by fitting a linear model (‘lm.fit‘) to the link-transformed response variable ‘g(y)‘. This provides a computationally cheap and stable way to initialize the main optimization algorithm.

Value

A named numeric vector containing the initial values for the model parameters (‘alpha‘, ‘varphi‘, ‘theta‘, ‘beta‘, ‘phi‘), ready to be used by an optimization routine.

Author(s)

Original code by Fabio M. Bayer (bayer@ufsm.br). Substantially modified and improved by Everton da Costa (everto.cost@gmail.com).

`summary.barma` *Summarize a barma Model Fit*

Description

(Full documentation block...)

Usage

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

Arguments

`object`	A fitted model object of class ‘"barma"‘.
`...`	Additional arguments (currently ignored).

Value

A list object of class ‘"summary.barma"‘...

Index