Package 'BTSR'

September 24, 2023

Type Package

Date 2023-09-22

Title Bounded Time Series Regression

Version 0.1.5

Copyright see file COPYRIGHTS

Depends R (>= 4.0.0)

Description Simulate, estimate and forecast a wide range of regression based dynamic models for bounded time series, covering the most commonly applied models in the literature. The main calculations are done in 'FORTRAN', which translates into very fast algorithms. The main references are
Bayer et al. (2017) <doi:10.1016/j.jhydrol.2017.10.006>,
Pumi et al. (2019) <doi:10.1016/j.jspi.2018.10.001>,
Pumi et al. (2021) <doi:10.1111/sjos.12439> and
Pumi et al. (2022) <arXiv:2211.02097>.

License GPL (>= 3)

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.2.3

Author Taiane Schaedler Prass [aut, cre, com]
(<https://orcid.org/0000-0003-3136-909X>),
Guilherme Pumi [ctb, aut] (<https://orcid.org/0000-0002-6256-3170>),
Fábio Mariano Bayer [ctb] (<https://orcid.org/0000-0002-1464-0805>),
Jack Joseph Dongarra [ctb],
Cleve Moler [ctb],
Gilbert Wright Stewart [ctb],
Ciyou Zhu [ctb],
Richard H. Byrd [ctb],
Jorge Nocedal [ctb],
Jose Luis Morales [ctb],
Peihuang Lu-Chen [ctb],
John Burkardt [ctb],
Alan Miller [ctb],
B.E. Schneider [ctb],

Alfred H. Morris [ctb],
D.E. Shaw [ctb],
Robert W.M. Wedderburn [ctb],
Jason Blevins [ctb],
Brian Wichman [ctb],
David Hill [ctb],
Hiroshi Takano [ctb],
George Marsaglia [ctb],
Jean-Michel Brankart [ctb],
Steve Kifowit [ctb],
Donald E. Knuth [ctb],
Catherine Loader [ctb]

Maintainer Taiane Schaedler Prass <taianeprass@gmail.com>

Repository CRAN

Date/Publication 2023-09-23 22:50:12 UTC

R topics documented:

`BARC.functions` *Functions to simulate, extract components and fit BARC models*

Description

These functions can be used to simulate, extract components and fit any model of the class `barc`. A model with class `barc` is a special case of a model with class `btsr` . See 'The BTSR structure' in `BARC.functions` for more details on the general structure. See 'Details'.

Usage

```
BARC.sim(n = 1, burn = 0, xreg = NULL, map = 4, coefs = list(alpha =
  0, beta = NULL, phi = NULL, theta = 0.5, nu = 20, u0 = pi/4),
  y.start = NULL, xreg.start = NULL, xregar = TRUE, error.scale = 0,
  complete = FALSE, linkg = c("linear", "linear"), linkh = "linear",
  ctt.h = 1, seed = NULL, rngtype = 2, debug = FALSE)

BARC.extract(yt, xreg = NULL, nnew = 0, xnew = NULL, p, r,
  coefs = list(), lags = list(), fixed.values = list(),
  fixed.lags = list(), y.start = NULL, xreg.start = NULL,
  xregar = TRUE, error.scale = 0, map = 4, linkg = c("linear",
  "linear"), linkh = "linear", ctt.h = 1, llk = TRUE, sco = FALSE,
  info = FALSE, debug = FALSE)

BARC.fit(yt, xreg = NULL, nnew = 0, xnew = NULL, p = 0, r = 1,
  start = list(), lags = list(), fixed.values = list(),
  ignore.start = FALSE, fixed.lags = list(), lower = list(nu = 0, u0 =
  0), upper = list(nu = Inf, u0 = 1), map = 4, linkg = c("linear",
  "linear"), linkh = "linear", ctt.h = 1, sco = FALSE, info = FALSE,
  xregar = TRUE, y.start = NULL, xreg.start = NULL, error.scale = 0,
  control = list(), report = TRUE, debug = FALSE, ...)
```

Arguments

n	a strictly positive integer. The sample size of yt (after burn-in). Default is 1.
burn	a non-negative integer. length of "burn-in" period. Default is 0.
xreg	optionally, a vector or matrix of external regressors. For simulation purposes, the length of xreg must be n+burn. Default is NULL. For extraction or fitting purposes, the length of xreg must be the same as the length of the observed time series y_t.
map	a non-negative integer from 1 to 5 corresponding to the map function. Default is 4. See 'The map function'.
coefs	a list with the coefficients of the model. An empty list will result in an error. The arguments that can be passed through this list are:

- alpha optionally, a numeric value corresponding to the intercept. If the argument is missing, it will be treated as zero. See 'The BTSR structure' in btsr.functions.
- beta optionally, a vector of coefficients corresponding to the regressors in xreg. If xreg is provided but beta is missing in the coefs list, an error message is issued.
- phi optionally, for the simulation function this must be a vector of size p, corresponding to the autoregressive coefficients (including the ones that are zero), where p is the AR order. For the extraction and fitting functions, this is a vector with the non-fixed values in the vector of autoregressive coefficients.

- `theta` the parameter (or vector of parameters) corresponding to the map function. If `map = 5` this value is ignored. For simulation, purposes, the default is `map = 4` and `theta = 0.5`.
- `nu` the dispersion parameter. If missing, an error message is issued.
- `u0` a numeric value in the interval $(0, 1)$, corresponding to the value of the random variable U_0. For simulation purposes, the default is `u0 = pi/4`.

`y.start` optionally, a initial value for yt (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $g_2(y_t) = 0$, for $t < 1$.

`xreg.start` optionally, a vector of initial value for xreg (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $X_t = 0$, for $t < 1$. If `xregar = FALSE` this argument is ignored.

`xregar` logical; indicates if xreg is to be included in the AR part of the model. See 'The BTSR structure'. Default is `TRUE`.

`error.scale` the scale for the error term. See 'The BTSR structure' in `btsr.functions`. Default is 0.

`complete` logical; if `FALSE` the function returns only the simulated time series yt, otherwise, additional time series are provided. Default is `FALSE`

`linkg` character or a two character vector indicating which links must be used in the model. See 'The BTSR structure' in `btsr.functions` for details and `link.btsr` for valid links. If only one value is provided, the same link is used for mu_t and for y_t in the AR part of the model. Default is `c("linear", "linear")`

`linkh` a character indicating which link must be associated to the the chaotic process. See 'The BTSR structure' in `btsr.functions` for details and `link.btsr` for valid links. Default is `"linear"`.

`ctt.h` numeric; the constant to be associated to the link h, when `linkh = "linear"`. Default is 1.

`seed` optionally, an integer which gives the value of the fixed seed to be used by the random number generator. If missing, a random integer is chosen uniformly from 1,000 to 10,000.

`rngtype` optionally, an integer indicating which random number generator is to be used. Default is 2. See 'Common Arguments' in `btsr.functions`.

`debug` logical, if `TRUE` the output from FORTRAN is return (for debuggin purposes). Default is `FALSE` for all models.

`yt` a numeric vector with the observed time series. If missing, an error message is issued.

`nnew` optionally, the number of out-of sample predicted values required. Default is 0.

`xnew` a vector or matrix, with `nnew` observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. If `xreg = NULL`, `xnew` is ignored.

`p` a non-negative integer. The order of AR polynomial. If missing, the value of `p` is calculated from length(coefs$phi) and length(fixed.values$phi). For fitting, the default is 0.

r	a non-negative integer. The size of the vector theta. If missing, the value of t is calculated from length(coefs$theta) and length(fixed.values$theta). For fitting, the default is 1.
lags	optionally, a list with the lags that the values in coefs correspond to. The names of the entries in this list must match the ones in coefs. For one dimensional coefficients, the lag is obviously always 1 and can be suppressed. An empty list indicates that either the argument fixed.lags is provided or all lags must be used.
fixed.values	optionally, a list with the values of the coefficients that are fixed. By default, if a given vector (such as the vector of AR coefficients) has fixed values and the corresponding entry in this list is empty, the fixed values are set as zero. The names of the entries in this list must match the ones in coefs.
fixed.lags	optionally, a list with the lags that the fixed values in fixed.values correspond to. The names of the entries in this list must match the ones in fixed.values. ##' For one dimensional coefficients, the lag is obviously always 1 and can be suppressed. If an empty list is provided and the model has fixed lags, the argument lags is used as reference.
llk	logical, if TRUE the value of the log-likelihood function is returned. Default is TRUE.
sco	logical, if TRUE the score vector is returned. Default is FALSE.
info	logical, if TRUE the information matrix is returned. Default is FALSE. For the fitting function, info is automatically set to TRUE when report = TRUE.
start	a list with the starting values for the non-fixed coefficients of the model. If an empty list is provided, the function coefs.start is used to obtain starting values for the parameters.
ignore.start	logical, if starting values are not provided, the function uses the default values and ignore.start is ignored. In case starting values are provided and ignore.start = TRUE, those starting values are ignored and recalculated. The default is FALSE.
lower	optionally, list with the lower bounds for the parameters. The names of the entries in these lists must match the ones in start. The default is to assume that the parameters have no lower bound except for nu, for which de default is 0. Only the bounds for bounded parameters need to be specified.
upper	optionally, list with the upper bounds for the parameters. The names of the entries in these lists must match the ones in start. The default is to assume that the parameters have no upper bound. Only the bounds for bounded parameters need to be specified.
control	a list with configurations to be passed to the optimization subroutines. Missing arguments will receive default values. See *fit.control*.
report	logical, if TRUE the summary from model estimation is printed and info is automatically set to TRUE. Default is TRUE.
...	further arguments passed to the internal functions.

Details

Neither the beta regression or an i.i.d. sample from a beta distribution can be obtained as special cases of the βARC model since the term $h(T(U_0))$ is always present

The model from Pumi et al. (2021) is obtained by setting `xregar = TRUE` (so that the regressors are included in the AR part of the model) and using the same link for y_t and μ_t.

The function `BARC.sim` generates a random sample from a βARC(p) model.

The function `BARC.extract` allows the user to extract the components y_t, μ_t, $\eta_t = g(\mu_t)$, r_t, $T^t(u_0)$, the log-likelihood, and the vectors and matrices used to calculate the score vector and the information matrix associated to a given set of parameters.

This function can be used by any user to create an objective function that can be passed to optimization functions not available in BTSR Package. At this point, there is no other use for which this function was intended.

The function `BARC.fit` fits a BARC model to a given univariate time series. For now, available optimization algorithms are `"L-BFGS-B"` and `"Nelder-Mead"`. Both methods accept bounds for the parameters. For `"Nelder-Mead"`, bounds are set via parameter transformation.

Value

The function `BARC.sim` returns the simulated time series yt by default. If `complete = TRUE`, a list with the following components is returned instead:

- `model`: string with the text `"BARC"`
- `yt`: the simulated time series
- `mut`: the conditional mean
- `etat`: the linear predictor $g(\mu_t)$
- `error`: the error term r_t
- `xreg`: the regressors (if included in the model).
- `debug`: the output from FORTRAN (if requested).

The function `BARC.extract` returns a list with the following components.

- `model`: string with the text `"BARC"`.
- `coefs`: the coefficients of the model passed through the `coefs` argument.
- `yt`: the observed time series.
- `gyt`: the transformed time series $g_2(y_t)$.
- `mut`: the conditional mean.
- `etat`: the linear predictor $g_1(\mu_t)$.
- `error`: the error term r_t.
- `xreg`: the regressors (if included in the model).
- `TS`: the chaotic process $T^t(u0)$.
- `sll`: the sum of the conditional log-likelihood (if requested).
- `sco`: the score vector (if requested).

- info: the information matrix (if requested).
- Drho, T, E, h: additional matrices and vectors used to calculate the score vector and the information matrix. (if requested).
- yt.new: the out-of-sample forecast (if requested).
- Ts.new: the out-of-sample forecast for the chaotic process (if requested).
- out.Fortran: FORTRAN output (if requested).

The function btsr.fit returns a list with the following components. Each particular model can have additional components in this list.

- model: string with the text "BARC"
- convergence: An integer code. 0 indicates successful completion. The error codes depend on the algorithm used.
- message: A character string giving any additional information returned by the optimizer, or NULL.
- counts: an integer giving the number of function evaluations.
- control: a list of control parameters.
- start: the starting values used by the algorithm.
- coefficients: The best set of parameters found.
- n: the sample size used for estimation.
- series: the observed time series
- gyt: the transformed time series $g_2(y_t)$
- fitted.values: the conditional mean, which corresponds to the in-sample forecast, also denoted fitted values
- etat: the linear predictor $g_1(\mu_t)$
- error.scale: the scale for the error term.
- error: the error term r_t
- residual: the observed minus the fitted values. The same as the error term if error.scale = 0.
- forecast: the out-of-sample forecast for y_t (if requested).
- Ts.forecas: the out-of-sample forecast for $T^t(u_0)$ (if requested).
- xnew: the observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. Only inlcudes if xreg is not NULL and nnew > 0.
- sll: the sum of the conditional log-likelihood (if requested)
- info.Matrix: the information matrix (if requested)
- configs: a list with the configurations adopted to fit the model. This information is used by the prediction function.
- out.Fortran: FORTRAN output (if requested)
- call: a string with the description of the fitted model.

The map function

The map function $T : [0, 1] \to [0, 1]$ is a dynamical system, i.e., a function, potentially depending on a r-dimensional vector of parameters θ. Available choices are

- map = 1, $\theta = k$, for k integer greater or equal to 2.

$$T(u) = (ku)(mod1)$$

- map = 2, $0 \leq \theta \leq 1$

$$T(u) = \frac{u}{\theta} I_(u < \theta) + \theta \frac{(u - \theta)}{(1 - \theta)} I(u \geq \theta)$$

- map = 3 (logistic map), $0 \leq \theta \leq 4$,

$$T(u) = \theta(1 - \theta)$$

- map = 4 (Manneville-Pomeau map), $0 < \theta < 1$

$$T(u) = (u + u^{1+\theta})(mod1)$$

- map = 5 (Lasota-Mackey's map),

$$T(u) = \frac{u}{(1 - u)} I(u \leq 0.5) + (2u - 1) I(u > 0.5)$$

References

Pumi, G.; Prass, T.S. and Souza, R.R. (2021). A dynamic model for double bounded time series with chaotic driven conditional averages. Scandinavian Journal of Statistics. Vol 48 (1), 68-86.

See Also

btsr.sim, btsr.extract, btsr.fit

btsr.extract

btsr.fit

Examples

```
m1 <- BARC.sim(linkg = "linear", linkh = "linear",
               n = 100, seed = 2021, complete = TRUE, ctt.h = 0.6,
              coefs = list(nu = 15, theta = 0.85, u0 = pi/4))

plot.ts(m1$yt)
lines(m1$mut, col = "red")

 #-------------------------------------------------------------
 # Generating a sample from a BARC model
 #-------------------------------------------------------------

 m1 <- BARC.sim(linkg = "linear", linkh = "linear",
```

```
              n = 100, seed = 2021, complete = TRUE, ctt.h = 0.6,
              coefs = list(nu = 15, theta = 0.85, u0 = pi/4))

#-------------------------------------------------------------
#  Extracting the conditional time series given yt and
#  a set of parameters
#-------------------------------------------------------------

 e1 = BARC.extract(yt = m1$yt, map = 4, ctt.h = 0.6,
                   coefs = list(nu = 15, theta = 0.85),
                   fixed.values = list(u0 = pi/4),
                   linkg = "linear", linkh = "linear", llk = TRUE,
                   sco = TRUE, info = TRUE)

#-----------------------------------------------------
# comparing the simulated and the extracted values
#-----------------------------------------------------
cbind(head(m1$mut), head(e1$mut))

#-------------------------------------------------------
# the log-likelihood, score vector and information matrix
# score vector and information matrix are obtained
# numerically.
#-------------------------------------------------------
e1$sll
e1$score
e1$info.Matrix


#-------------------------------------------------------------
# Generating a sample from a BARC model
#-------------------------------------------------------------

m1 <- BARC.sim(linkg = "linear", linkh = "linear",
              n = 100, seed = 2021, complete = TRUE, ctt.h = 0.6,
              coefs = list(nu = 15, theta = 0.85, u0 = pi/4))

#-------------------------------------------------------------
#  Fitting a BARC model. Assuming only alpha fixed.
#-------------------------------------------------------------
 f1 = BARC.fit(yt = m1$yt, map = 4, ctt.h = 0.6,
               start = list(nu = 10, theta = 0.6, u0 = 0.5),
               lower = list(nu = 0, theta = 0, u0 = 0),
               upper = list(theta = 1, u0 = 1),
               fixed.values = list(alpha = 0),
               control = list(iprint = -1, method = "Nelder-Mead"))

 coefficients(f1)

 plot.ts(m1$yt)
 lines(f1$fitted.values, col = "red")

#-------------------------------------------------------------
```

```
#  Out-of-sample forecast
#------------------------------------------------------------
pred = predict(f1, nnew = 5)
pred$forecast
pred$Ts.forecast
```

BARFIMA.functions *Functions to simulate, extract components and fit BARFIMA models*

Description

These functions can be used to simulate, extract components and fit any model of the class barfima. A model with class barfima is a special case of a model with class btsr . See 'The BTSR structure' in btsr.functions for more details on the general structure.

The βARMA model, the beta regression and a i.i.d. sample from a beta distribution can be obtained as special cases. See 'Details'.

Usage

```
BARFIMA.sim(n = 1, burn = 0, xreg = NULL, coefs = list(alpha = 0, beta
  = NULL, phi = NULL, theta = NULL, d = 0, nu = 20), y.start = NULL,
  xreg.start = NULL, xregar = TRUE, error.scale = 1, complete = FALSE,
  inf = 1000, linkg = c("logit", "logit"), seed = NULL, rngtype = 2,
  debug = FALSE)

BARFIMA.extract(yt, xreg = NULL, nnew = 0, xnew = NULL, p, q,
  coefs = list(), lags = list(), fixed.values = list(),
  fixed.lags = list(), y.start = NULL, xreg.start = NULL,
  xregar = TRUE, error.scale = 1, inf = 1000, m = 0,
  linkg = c("logit", "logit"), llk = TRUE, sco = FALSE, info = FALSE,
  extra = FALSE, debug = FALSE)

BARFIMA.fit(yt, xreg = NULL, nnew = 0, xnew = NULL, p = 0, d = TRUE,
  q = 0, m = 0, inf = 1000, start = list(), ignore.start = FALSE,
  lags = list(), fixed.values = list(), fixed.lags = list(),
  lower = list(nu = 0), upper = list(nu = Inf), linkg = c("logit",
  "logit"), sco = FALSE, info = FALSE, extra = FALSE, xregar = TRUE,
  y.start = NULL, xreg.start = NULL, error.scale = 1, control = list(),
  report = TRUE, debug = FALSE, ...)
```

Arguments

n	a strictly positive integer. The sample size of yt (after burn-in). Default is 1.
burn	a non-negative integer. The length of the "burn-in" period. Default is 0.

`xreg` optionally, a vector or matrix of external regressors. For simulation purposes, the length of xreg must be `n+burn`. Default is `NULL`. For extraction or fitting purposes, the length of `xreg` must be the same as the length of the observed time series y_t.

`coefs` a list with the coefficients of the model. An empty list will result in an error. The arguments that can be passed through this list are:

- `alpha` optionally, a numeric value corresponding to the intercept. If the argument is missing, it will be treated as zero. See 'The BTSR structure' in `btsr.functions`.
- `beta` optionally, a vector of coefficients corresponding to the regressors in `xreg`. If `xreg` is provided but `beta` is missing in the `coefs` list, an error message is issued.
- `phi` optionally, for the simulation function this must be a vector of size p, corresponding to the autoregressive coefficients (including the ones that are zero), where p is the AR order. For the extraction and fitting functions, this is a vector with the non-fixed values in the vector of autoregressive coefficients.
- `theta` optionally, for the simulation function this must be a vector of size q, corresponding to the moving average coefficients (including the ones that are zero), where q is the MA order. For the extraction and fitting functions, this is a vector with the non-fixed values in the vector of moving average coefficients.
- `d` optionally, a numeric value corresponding to the long memory parameter. If the argument is missing, it will be treated as zero.
- `nu` the dispersion parameter. If missing, an error message is issued.

`y.start` optionally, an initial value for yt (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $g_2(y_t) = 0$, for $t < 1$.

`xreg.start` optionally, a vector of initial value for xreg (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $X_t = 0$, for $t < 1$. If `xregar = FALSE` this argument is ignored.

`xregar` logical; indicates if xreg is to be included in the AR part of the model. See 'The BTSR structure'. Default is `TRUE`.

`error.scale` the scale for the error term. See 'The BTSR structure' in `btsr.functions`. Default is 1.

`complete` logical; if `FALSE` the function returns only the simulated time series yt, otherwise, additional time series are provided. Default is `FALSE`

`inf` the truncation point for infinite sums. Default is 1,000. In practice, the Fortran subroutine uses $inf = q$, if $d = 0$.

`linkg` character or a two character vector indicating which links must be used in the model. See 'The BTSR structure' in `btsr.functions` for details and `link.btsr` for valid links. If only one value is provided, the same link is used for mu_t and for y_t in the AR part of the model. Default is `c("logit", "logit")`. For the linear link, the constant will be always 1.

`seed` optionally, an integer which gives the value of the fixed seed to be used by the random number generator. If missing, a random integer is chosen uniformly from 1,000 to 10,000.

rngtype	optionally, an integer indicating which random number generator is to be used. Default is 2: the Mersenne Twister algorithm. See 'Common Arguments' in btsr.functions.
debug	logical, if TRUE the output from FORTRAN is return (for debugging purposes). Default is FALSE for all models.
yt	a numeric vector with the observed time series. If missing, an error message is issued.
nnew	optionally, the number of out-of sample predicted values required. Default is 0.
xnew	a vector or matrix, with nnew observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. If xreg = NULL, xnew is ignored.
p	a non-negative integer. The order of AR polynomial. If missing, the value of p is calculated from length(coefs$phi) and length(fixed.values$phi). For fitting, the default is 0.
q	a non-negative integer. The order of the MA polynomial. If missing, the value of q is calculated from length(coefs$theta) and length(fixed.values$theta). For fitting, the default is 0.
lags	optionally, a list with the lags that the values in coefs correspond to. The names of the entries in this list must match the ones in coefs. For one dimensional coefficients, the lag is obviously always 1 and can be suppressed. An empty list indicates that either the argument fixed.lags is provided or all lags must be used.
fixed.values	optionally, a list with the values of the coefficients that are fixed. By default, if a given vector (such as the vector of AR coefficients) has fixed values and the corresponding entry in this list is empty, the fixed values are set as zero. The names of the entries in this list must match the ones in coefs.
fixed.lags	optionally, a list with the lags that the fixed values in fixed.values correspond to. The names of the entries in this list must match the ones in fixed.values. ##' For one dimensional coefficients, the lag is obviously always 1 and can be suppressed. If an empty list is provided and the model has fixed lags, the argument lags is used as reference.
m	a non-negative integer indicating the starting time for the sum of the partial log-likelihoods, that is $\ell = \sum_{t=m+1}^{n} \ell_t$. Default is 0.
llk	logical, if TRUE the value of the log-likelihood function is returned. Default is TRUE.
sco	logical, if TRUE the score vector is returned. Default is FALSE.
info	logical, if TRUE the information matrix is returned. Default is FALSE. For the fitting function, info is automatically set to TRUE when report = TRUE.
extra	logical, if TRUE the matrices and vectors used to calculate the score vector and the information matrix are returned. Default is FALSE.
d	logical, if TRUE, the parameter d is included in the model either as fixed or non-fixed. If d = FALSE the value is fixed as 0. The default is TRUE.
start	a list with the starting values for the non-fixed coefficients of the model. If an empty list is provided, the function coefs.start is used to obtain starting values for the parameters.

ignore.start	logical, if starting values are not provided, the function uses the default values and ignore.start is ignored. In case starting values are provided and ignore.start = TRUE, those starting values are ignored and recalculated. The default is FALSE.
lower	optionally, list with the lower bounds for the parameters. The names of the entries in these lists must match the ones in start. The default is to assume that the parameters have no lower bound except for nu, for which de default is 0. Only the bounds for bounded parameters need to be specified.
upper	optionally, list with the upper bounds for the parameters. The names of the entries in these lists must match the ones in start. The default is to assume that the parameters have no upper bound. Only the bounds for bounded parameters need to be specified.
control	a list with configurations to be passed to the optimization subroutines. Missing arguments will receive default values. See *fit.control*.
report	logical, if TRUE the summary from model estimation is printed and info is automatically set to TRUE. Default is TRUE.
...	further arguments passed to the internal functions.

Details

The βARMA model and the beta regression can be obtained as special cases of the βARFIMA model.

- βARFIMA: the model from Pumi et al. (2019) is obtained by setting error.scale = 1 (predictive scale) and xregar = TRUE (so that the regressors are included in the AR part of the model). Variations of this model are obtained by changing error.scale, xregar and/or by using different links for $y[t]$ (in the AR part of the model) and $\mu[t]$.
- βARMA: the model from Rocha and Cribari-Neto (2009, 2017) is obtained by setting coefs$d = 0 and d = FALSE and error.scale = 1 (predictive scale). Variations of this model are obtained by changing the error scale and/or by using a different link for $y[t]$ in the AR part of the model.
- beta regression: the model from Ferrari and Cribari-Neto (2004) is obtained by setting p = 0, q = 0 and coefs$d = 0 and d = FALSE. The error.scale is irrelevant. The second argument in linkg is irrelevant.
- an i.i.d. sample from a Beta distribution with parameters shape1 and shape2 (compatible with the one from rbeta) is obtained by setting linkg = "linear", p = 0, q = 0, d = FALSE and, in the coefficient list, alpha = shape1/(shape1+shape2) and nu = shape1+shape2. (error.scale and xregar are irrelevant)

The function BARFIMA.sim generates a random sample from a βARFIMA(p,d,q) model.

The function BARFIMA.extract allows the user to extract the components y_t, μ_t, $\eta_t = g(\mu_t)$, r_t, the log-likelihood, and the vectors and matrices used to calculate the score vector and the information matrix associated to a given set of parameters.

This function can be used by any user to create an objective function that can be passed to optimization algorithms not available in the BTSR Package.

The function BARFIMA.fit fits a BARFIMA model to a given univariate time series. For now, available optimization algorithms are "L-BFGS-B" and "Nelder-Mead". Both methods accept bounds for the parameters. For "Nelder-Mead", bounds are set via parameter transformation.

Value

The function BARFIMA.sim returns the simulated time series yt by default. If complete = TRUE, a list with the following components is returned instead:

- model: string with the text "BARFIMA"
- yt: the simulated time series
- mut: the conditional mean
- etat: the linear predictor $g(\mu_t)$
- error: the error term r_t
- xreg: the regressors (if included in the model).
- debug: the output from FORTRAN (if requested).

The function BARFIMA.extract returns a list with the following components.

- model: string with the text "BARFIMA"
- coefs: the coefficients of the model passed through the coefs argument
- yt: the observed time series
- gyt: the transformed time series $g_2(y_t)$
- mut: the conditional mean
- etat: the linear predictor $g_1(\mu_t)$
- error: the error term r_t
- xreg: the regressors (if included in the model).
- sll: the sum of the conditional log-likelihood (if requested)
- sco: the score vector (if requested)
- info: the information matrix (if requested)
- Drho, T, E, h: additional matrices and vectors used to calculate the score vector and the information matrix. (if requested)
- yt.new: the out-of-sample forecast (if requested)
- out.Fortran: FORTRAN output (if requested)

The function btsr.fit returns a list with the following components. Each particular model can have additional components in this list.

- model: string with the text "BARFIMA"
- convergence: An integer code. 0 indicates successful completion. The error codes depend on the algorithm used.
- message: A character string giving any additional information returned by the optimizer, or NULL.

- `counts`: an integer giving the number of function evaluations.
- `control`: a list of control parameters.
- `start`: the starting values used by the algorithm.
- `coefficients`: The best set of parameters found.
- `n`: the sample size used for estimation.
- `series`: the observed time series
- `gyt`: the transformed time series $g_2(y_t)$
- `fitted.values`: the conditional mean, which corresponds to the in-sample forecast, also denoted fitted values
- `etat`: the linear predictor $g_1(\mu_t)$
- `error.scale`: the scale for the error term.
- `error`: the error term r_t
- `residual`: the observed minus the fitted values. The same as the `error` term if `error.scale = 0`.
- `forecast`: the out-of-sample forecast (if requested).
- `xnew`: the observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. Only inlcudes if `xreg` is not `NULL` and `nnew > 0`.
- `sll`: the sum of the conditional log-likelihood (if requested)
- `info.Matrix`: the information matrix (if requested)
- `configs`: a list with the configurations adopted to fit the model. This information is used by the prediction function.
- `out.Fortran`: FORTRAN output (if requested)
- `call`: a string with the description of the fitted model.

References

Ferrari, S.L.P. and Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions. J. Appl. Stat. 31 (7), 799-815.

Pumi, G.; Valk, M.; Bisognin, C.; Bayer, F.M. and Prass, T.S. (2019). Beta autoregressive fractionally integrated moving average models. Journal of Statistical Planning and Inference (200), 196-212.

Rocha, A.V. and Cribari-Neto, F. (2009). Beta autoregressive moving average models. Test 18 (3), 529–545.

Rocha, A.V. and Cribari-Neto, F. (2017). Erratum to: Beta autoregressive moving average models. Test 26 (2), 451-459.

See Also

`btsr.sim`

`btsr.extract`

`btsr.fit`

Examples

```
# Generating a Beta model were mut does not vary with time
# yt ~ Beta(a,b), a = mu*nu, b = (1-mu)*nu

y <- BARFIMA.sim(linkg = "linear", n = 1000, seed = 2021,
                 coefs = list(alpha = 0.2, nu = 20))
hist(y)

 #-------------------------------------------------------------
 # Generating a Beta model were mut does not vary with time
 # yt ~ Beta(a,b), a = mu*nu, b = (1-mu)*nu
 #-------------------------------------------------------------

 m1 <- BARFIMA.sim(linkg = "linear",n = 100,
                   complete = TRUE, seed = 2021,
                   coefs = list(alpha = 0.2, nu = 20))

 #-------------------------------------------------------------
 #  Extracting the conditional time series given yt and
 #  a set of parameters
 #-------------------------------------------------------------

 # Assuming that all coefficients are non-fixed
 e1 = BARFIMA.extract(yt = m1$yt, coefs = list(alpha = 0.2, nu = 20),
                      link = "linear", llk = TRUE,
                      sco = TRUE, info = TRUE)

 #-----------------------------------------------------
 # comparing the simulated and the extracted values
 #-----------------------------------------------------
 cbind(head(m1$mut), head(e1$mut))

 #-------------------------------------------------------------
 # the log-likelihood, score vector and information matrix
 #-------------------------------------------------------------
 e1$sll
 e1$score
 e1$info.Matrix


# Generating a Beta model were mut does not vary with time
# yt ~ Beta(a,b), a = mu*nu, b = (1-mu)*nu

y <- BARFIMA.sim(linkg = "linear", n = 100, seed = 2021,
                 coefs = list(alpha = 0.2, nu = 20))

# fitting the model
f <- BARFIMA.fit(yt = y, report = TRUE,
                 start = list(alpha = 0.5, nu = 10),
                 linkg = "linear", d = FALSE)
```

`btsr.functions` *Generic functions to simulate, extract components and fit BTSR models*

Description

These generic functions can be used to simulate, extract components and fit any model of the class `btsr`. All functions are wrappers for the corresponding function associated to the chosen model. See 'The BTSR structure' and 'Common Arguments'.

Usage

```
btsr.sim(model, complete = FALSE, ...)

btsr.extract(model, ...)

btsr.fit(model, ...)
```

Arguments

`model`	character; one of `"BARFIMA"`, `"GARFIMA"`, `"KARFIMA"`, `"BARC"`.
`complete`	logical; if `FALSE` the function returns only the simulated time series yt, otherwise, additional time series are provided. Default is `FALSE` for all models.
`...`	further arguments passed to the functions, according to the model selected in the argument `model`. See 'Common Arguments'

Details

The function `btsr.sim` is used to generate random samples from BTSR models. See 'The BTSR structure'.

The function `btsr.extract` allows the user to extract the components y_t, μ_t, $\eta_t = g(\mu_t)$, r_t, the log-likelihood, and the vectors and matrices used to calculate the score vector and the information matrix associated to a given set of parameters.

This function can be used by any user to create an objective function that can be passed to optimization functions not available in BTSR Package. At this point, there is no other use for which this function was intended.

The function `btsr.fit` fits a BTSR model to a given univariate time series. For now, available optimization algorithms are `"L-BFGS-B"` and `"Nelder-Mead"`. Both methods accept bounds for the parameters. For `"Nelder-Mead"`, bounds are set via parameter transformation.

Value

The function `btsr.sim` returns the simulated time series yt by default. If `complete = TRUE`, a list with the following components is returned instead:

- `model`: character; one of `"BARFIMA"`, `"GARFIMA"`, `"KARFIMA"`, `"BARC"`. (same as the input argument)

- yt: the simulated time series
- gyt: the transformed time series $g2(y_t)$
- mut: the conditional mean
- etat: the linear predictor $g(\mu_t)$
- error: the error term r_t
- xreg: the regressors (if included in the model).
- debug: the output from FORTRAN (if requested).

The function btsr.extract returns a list with the following components. Each particular model can have additional components in this list.

- model: character; one of "BARFIMA", "GARFIMA", "KARFIMA", "BARC". (same as the input argument)
- coefs: the coefficients of the model passed through the coefs argument
- yt: the observed time series
- gyt: the transformed time series $g_2(y_t)$
- mut: the conditional mean
- etat: the linear predictor $g_1(\mu_t)$
- error: the error term r_t
- xreg: the regressors (if included in the model).
- forecast: the out-of-sample forecast (if requested).
- xnew: the observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. Only inlcudes if xreg is not NULL and nnew > 0.
- sll: the sum of the conditional log-likelihood (if requested)
- sco: the score vector (if requested)
- info: the information matrix (if requested)
- Drho, T, E, h: additional matrices and vectors used to calculate the score vector and the information matrix. (if requested)
- yt.new: the out-of-sample forecast (if requested)
- out.Fortran: FORTRAN output (if requested)

The function btsr.fit returns a list with the following components. Each particular model can have additional components in this list.

- model: character; one of "BARFIMA", "GARFIMA", "KARFIMA", "BARC". (same as the input argument)
- convergence: An integer code. 0 indicates successful completion. The error codes depend on the algorithm used.
- message: A character string giving any additional information returned by the optimizer, or NULL.
- counts: an integer giving the number of function evaluations.
- control: a list of control parameters.

- `start`: the starting values used by the algorithm.
- `coefficients`: The best set of parameters found.
- `n`: the sample size used for estimation.
- `series`: the observed time series
- `gyt`: the transformed time series $g_2(y_t)$
- `fitted.values`: the conditional mean, which corresponds to the in-sample forecast, also denoted fitted values
- `etat`: the linear predictor $g_1(\mu_t)$
- `error.scale`: the scale for the error term.
- `error`: the error term r_t
- `residuals`: the observed minus the fitted values. The same as the `error` term if `error.scale = 0`.
- `sll`: the sum of the conditional log-likelihood (if requested)
- `info.Matrix`: the information matrix (if requested)
- `configs`: a list with the configurations adopted to fit the model. This information is used by the prediction function.
- `out.Fortran`: FORTRAN output (if requested)
- `call`: a string with the description of the fitted model.

The BTSR structure

The general structure of the deterministic part of a BTSR model is

$$g_1(\mu_t) = \alpha + X_t\beta + \sum_{j=1}^{p} \phi_j[g_2(y_{t-j}) - I_{xregar}X_{t-j}\beta] + h_t$$

where

- I_{xregar} is 0, if `xreg` is not included in the AR part of the model and 1, otherwise
- the term h_t depends on the argument `model`:
  - for BARC models: $h_t = h(T^{t-1}(u_0))$
  - otherwise: $h_t = \sum_{k=1}^{\infty} c_k r_{t-k}$
- g_1 and g_2 are the links defined in `linkg`. Notice that g_2 is only used in the AR part of the model and, typically, $g_1 = g_2$.
- r_t depends on the `error.scale` adopted:
  - if `error.scale = 0`: $r_t = y_t - \mu_t$ (data scale)
  - if `error.scale = 1`: $r_t = g_1(y_t) - g_1(\mu_t)$ (predictive scale)
- c_k are the coefficients of $(1-L)^d\theta(L)$. In particular, if $d = 0$, then $c_k = \theta_k$, for $k = 1, \ldots, q$.

Common Arguments

In what follows we describe some of the arguments that are commom to all BTSR models. For more details on extra arguments, see the corresponding function associated to the selected model.

Simulation Function:

Common arguments passed through `"..."` in `btsr.sim` are:

- `n` a strictly positive integer. The sample size of yt (after burn-in). Default for all models is 1.
- `burn` a non-negative integer. length of "burn-in" period. Default for all models is 0.
- `xreg` optionally, a vector or matrix of external regressors. For simulation purposes, the length of xreg must be `n+burn`. Default for all models is `NULL`
- `coefs` a list with the coefficients of the model. Each model has its default. An empty list will result in an error. The arguments in this list are:
  - `alpha` optionally, A numeric value corresponding to the intercept. If the argument is missing, it will be treated as zero.
  - `beta` optionally, a vector of coefficients corresponding to the regressors in `xreg`. If `xreg` is provided but `beta` is missing in the `coefs` list, an error message is issued.
  - `phi` optionally, a vector of size p, corresponding to the autoregressive coefficients (including the ones that are zero), where p is the AR order.
  - `nu` the dispersion parameter. If missing, an error message is issued.
  - `rho, y.lower, y.upper, theta, d, u0` model specif arguments. See the documentation corresponding to each model.
- `y.start` optionally, a initial value for yt (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $g_2(y_t) = 0$, for $t < 1$.
- `xreg.start` optionally, a vector of initial value for xreg (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $X_t = 0$, for $t < 1$. If `xregar = FALSE` this argument is ignored.
- `xregar` logical; indicates if xreg is to be included in the AR part of the model. See 'The BTSR structure'. Default is `TRUE`.
- `error.scale` the scale for the error term. See also 'The BTSR structure'. Each model has its default.
- `inf` the truncation point for infinite sums. Default is 1000. In practice, the Fortran subroutine uses $inf = q$, if $d = 0$. BARC models do not have this argument.
- `link` character or a two character vector indicating which links must be used in the model. See 'The BTSR structure'. If only one value is provided, the same link is used for mu_t and for y_t in the AR part of the model. Each model has its default.
- `seed` optionally, an integer which gives the value of the fixed seed to be used by the random number generator. If missing, a random integer is chosen uniformly from 1,000 to 10,000.
- `rngtype` optionally, an integer indicating which random number generator is to be used. Default is 2. The current options are:
  - `0`: Jason Blevins algorithm. Available at https://jblevins.org/log/openmp
  - `1`: Wichmann-Hill algorithm (Wichmann and Hill, 1982).
  - `2`: Mersenne Twister algorithm (Matsumoto and Nishimura, 1998). FORTRAN code adapted from https://jblevins.org/mirror/amiller/mt19937.f90 and https://jblevins.org/mirror/amiller/mt19937a.f90

  - 3: Marsaglia-MultiCarry algorithm - kiss 32. Random number generator suggested by George Marsaglia in "Random numbers for C: The END?" posted on sci.crypt.random-numbers in 1999.
  - 4: Marsaglia-MultiCarry algorithm - kiss 64. Based on the 64-bit KISS (Keep It Simple Stupid) random number generator distributed by George Marsaglia in https://groups.google.com/d/topic/comp.lang.fortran/qFv18ql_WlU
  - 5: Knuth's 2002 algorithm (Knuth, 202). FORTRAN code adapted from https://www-cs-faculty.stanford.edu/~knuth/programs/frng.f
  - 6: L'Ecuyer's 1999 algorithm - 64-bits (L'Ecuyer, 1999). FORTRAN code adapted from https://jblevins.org/mirror/amiller/lfsr258.f90

  For more details on these algorithms see Random and references therein.
- debug logical, if TRUE the output from FORTRAN is return (for debuggin purposes). Default is FALSE for all models.

Extracting Function:

Common arguments passed through "..." in btsr.extract are:

- yt a numeric vector with the observed time series. If missing, an error message is issued.
- xreg optionally, a vector or matrix with the regressor's values. Default is NULL for all models.
- nnew optionally, the number of out-of sample predicted values required. Default is 0 for all models.
- xnew a vector or matrix, with nnew observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. If xreg = NULL, xnew is ignored.
- p a non-negative integer. The order of AR polynomial. If missing, the value of p is calculated from length(coefs$phi) and length(fixed.values$phi).
- q,r a non-negative integer. The order of the MA polynomial and the size of the vector of parameters for the map function (BARC only). If missing, the argument is calcualted based on length(coefs$theta) and length(fixed.values$theta).
- coefs a list with the coefficients of the model. Each model has its default. Passing both, coefs and fixed.values empty will result in an error. The arguments in this list are
  - alpha a numeric value corresponding to the intercept. If missing, will be set as zero.
  - beta a vector of coefficients corresponding to the regressors in xreg. If xreg is provided but beta is missing in the coefs list, an error message is issued.
  - phi a vector with the non-fixed values in the vector of AR coefficients.
  - nu the dispersion parameter. If missing, an error message is issued.
  - theta, d, u0 model specific arguments. See the documentation corresponding to each model.
- lags optionally, a list with the lags that the values in coefs correspond to. The names of the entries in this list must match the ones in coefs. For one dimensional coefficients, the lag is obviously always 1 and can be suppressed. An empty list indicates that either the argument fixed.lags is provided or all lags must be used.
- fixed.values optionally, a list with the values of the coefficients that are fixed. By default, if a given vector (such as the vector of AR coefficients) has fixed values and the corresponding entry in this list is empty, the fixed values are set as zero. The names of the entries in this list must match the ones in coefs.

- `fixed.lags` optionally, a list with the lags that the fixed values in `fixed.values` correspond to. The names of the entries in this list must match the ones in `fixed.values`. ##' For one dimensional coefficients, the `lag` is obviously always 1 and can be suppressed. If an empty list is provided and the model has fixed lags, the argument `lags` is used as reference.
- `y.start` optionally, a initial value for yt (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $g_2(y_t) = 0$, for $t < 1$.
- `xreg.start` optionally, a vector of initial value for xreg (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $X_t = 0$, for $t < 1$. If `xregar = FALSE` this argument is ignored.
- `xregar` logical; indicates if xreg is to be included in the AR part of the model. See 'The BTSR structure'. Default is `TRUE`.
- `error.scale` the scale for the error term. See also 'The BTSR structure'. Each model has its default.
- `inf` the truncation point for infinite sums. Default is 1. BARC models do not have this argument.
- `m` a non-negative integer indicating the starting time for the sum of the partial log-likelihoods, that is $\ell = \sum_{t=m+1}^{n} \ell_t$. Default is 0.
- `linkg` character or a two character vector indicating which links must be used in the model. See 'The BTSR structure'. If only one value is provided, the same link is used for mu_t and for y_t in the AR part of the model. Each model has its default.
- `llk` logical, if `TRUE` the value of the log-likelihood function is returned. Default is `TRUE` for all models.
- `sco` logical, if `TRUE` the score vector is returned. Default is `FALSE` for all models.
- `info` logical, if `TRUE` the information matrix is returned. Default is `FALSE` for all models.
- `extra` logical, if `TRUE` the matrices and vectors used to calculate the score vector and the information matrix are returned. Default is `FALSE` for all models.
- `debug` logical, if `TRUE` the output from FORTRAN is return (for debuggin purposes). Default is `FALSE` for all models.

Fitting Function:

Common arguments passed through `"..."` in `btsr.fit` are the same as in `btsr.extract` plus the following:

- `d` logical, if `TRUE`, the parameter `d` is included in the model either as fixed or non-fixed. If `d = FALSE` the value is fixed as 0. The default is `TRUE` for all models, except BARC that does not have this parameter.
- `start` a list with the starting values for the non-fixed coefficients of the model. If an empty list is provided, the function `coefs.start` is used to obtain starting values for the parameters.
- `ignore.start` logical, if starting values are not provided, the function uses the default values and `ignore.start` is ignored. In case starting values are provided and `ignore.start = TRUE`, those starting values are ignored and recalculated. The default is `FALSE`.
- `lower, upper` optionally, list with the lower and upper bounds for the parameters. The names of the entries in these lists must match the ones in `start`. The default is to assume that the parameters are unbounded. Only the bounds for bounded parameters need to be specified.
- `control` a list with configurations to be passed to the optimization subroutines. Missing arguments will receive default values. See *fit.control*.
- `report` logical, if `TRUE` the summary from model estimation is printed and `info` is automatically set to `TRUE`. Default is `TRUE`.

References

Knuth, D. E. (2002). The Art of Computer Programming. Volume 2, third edition, ninth printing.

L'Ecuyer, P. (1999). Good parameters and implementations for combined multiple recursive random number generators. Operations Research, 47, 159-164. doi:10.1287/opre.47.1.159.

Matsumoto, M. and Nishimura, T. (1998). Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Transactions on Modeling and Computer Simulation, 8, 3-30.

Wichmann, B. A. and Hill, I. D. (1982). Algorithm AS 183: An Efficient and Portable Pseudo-random Number Generator. Applied Statistics, 31, 188-190; Remarks: 34, 198 and 35, 89. doi:10.2307/2347988.

See Also

BARFIMA.sim, GARFIMA.sim, KARFIMA.sim, BARC.sim

BARFIMA.extract, GARFIMA.extract, KARFIMA.extract, BARC.extract

BARFIMA.fit, GARFIMA.fit, KARFIMA.fit, BARC.fit

Examples

```
# Generating a Beta model were mut does not vary with time
# yt ~ Beta(a,b), a = mu*nu, b = (1-mu)*nu

y <- btsr.sim(model= "BARFIMA", linkg = "linear",
              n = 1000, seed = 2021,
              coefs = list(alpha = 0.2, nu = 20))
hist(y)

 #-------------------------------------------------------------
 # Generating a Beta model were mut does not vary with time
 # yt ~ Beta(a,b), a = mu*nu, b = (1-mu)*nu
 #-------------------------------------------------------------

 m1 <- btsr.sim(model= "BARFIMA", linkg = "linear",
                n = 100, seed = 2021, complete = TRUE,
                coefs = list(alpha = 0.2, nu = 20))

 #-------------------------------------------------------------
 #  Extracting the conditional time series given yt and
 #  a set of parameters
 #-------------------------------------------------------------

 # Assuming that all coefficients are non-fixed
 e1 = btsr.extract(model = "BARFIMA", yt = m1$yt,
                   coefs = list(alpha = 0.2, nu = 20),
                   link = "linear", llk = TRUE,
                   sco = TRUE, info = TRUE)

 # Assuming that all coefficients are fixed
 e2 = btsr.extract(model = "BARFIMA", yt = m1$yt,
```

```
                  fixed.values = list(alpha = 0.2, nu = 20),
                  link = "linear", llk = TRUE,
                  sco = TRUE, info = TRUE)

# Assuming at least one fixed coefficient and one non-fixed
e3 = btsr.extract(model = "BARFIMA", yt = m1$yt,
                  fixed.values = list(alpha = 0.2, nu = 20),
                  link = "linear", llk = TRUE,
                  sco = TRUE, info = TRUE)
e4 = btsr.extract(model = "BARFIMA", yt = m1$yt,
                  fixed.values = list(alpha = 0.2, nu = 20),
                  link = "linear", llk = TRUE,
                  sco = TRUE, info = TRUE)

#-----------------------------------------------------
# comparing the simulated and the extracted values
#-----------------------------------------------------
cbind(head(m1$mut), head(e1$mut), head(e2$mut), head(e3$mut), head(e4$mut))

#-----------------------------------------------------
# comparing the log-likelihood values obtained (must be the all equal)
#-----------------------------------------------------
c(e1$sll, e2$sll, e3$sll, e4$sll)

#-----------------------------------------------------
# comparing the score vectors:
#-----------------------------------------------------
# - e1 must have 2 values: dl/dmu and dl/dnu
# - e2 must be empty
# - e3 and e4 must have one value corresponding
#    to the non-fixed coefficient
#-----------------------------------------------------
 e1$score
 e2$score
 e3$score
 e4$score

#-----------------------------------------------------
# comparing the information matrices.
#-----------------------------------------------------
# - e1 must be a 2x2 matrix
# - e2 must be empty
# - e3 and e4 must have one value corresponding
#    to the non-fixed coefficient
#-----------------------------------------------------
 e1$info.Matrix
 e2$info.Matrix
 e3$info.Matrix
 e4$info.Matrix


# Generating a Beta model were mut does not vary with time
# yt ~ Beta(a,b), a = mu*nu, b = (1-mu)*nu
```

```
y <- btsr.sim(model= "BARFIMA", linkg = "linear",
              n = 100, seed = 2021,
              coefs = list(alpha = 0.2, nu = 20))

# fitting the model
f <- btsr.fit(model = "BARFIMA", yt = y, report = TRUE,
             start = list(alpha = 0.5, nu = 10),
             linkg = "linear", d = FALSE)
```

`coefs.start` *Initial values for optimization*

Description

This function calculates initial values for the parameter vector, to pass to the optimization function.

Usage

```
coefs.start(model = "Generic", yt, linkg = c("linear", "linear"),
  xreg = NULL, p = 0, q = 0, d = TRUE, y.start = NULL,
  y.lower = -Inf, y.upper = Inf, lags = list(), fixed.values = list(),
  fixed.lags = list())
```

Arguments

`model`	character; The model to be fitted to the data. One of `"BARFIMA"`, `"KARFIMA"`, `"GARFIMA"`, `"BARC"`. Default is "Generic" so that no specific structure is assumed.
`yt`	a univariate time series. Missing values (NA's) are not allowed.
`linkg`	character; one of `"linear"`, `"logit"`, `"log"`, `"loglog"`, `"cloglog"`. If only one name is provided, the same link will be used for the conditional mean, that is to define $g(\mu)$ and for the observed time series in the AR part of the model, that is, $g(y[t])$.
`xreg`	optional; a vector or matrix of external regressors, which must have the same number of rows as x.
`p`	an integer; the AR order. Default is zero.
`q`	an integer; for `BARC` models represents the dimension of the parameter associated to the map T. For other models is the MA order. Default is zero.
`d`	logical; if FALSE, d is fixed as zero. Default is TRUE.
`y.start`	optional; an initialization value for $y[t]$, for $t \leq 0$, to be used in the AR recursion. If not provided, the default assume $y[t] = 0$, when using a `"linear"` link for yt, and $g(y[t]) = 0$, otherwise.
`y.lower`	lower limit for the distribution support. Default is `-Inf`.

y.upper	upper limit for the distribution support. Default is Inf.
lags	optional; a list with the components beta, phi and theta specifying which lags must be included in the model. An empty list or missing component indicates that, based on the values nreg, p e q), all lags must be includes in the model.
fixed.values	optional; a list with the fixed values for each component, if any. If fixed values are provided, either lags or fixed.lags must also be provided.
fixed.lags	optional; a list with the components beta, phi and theta specifying which lags must be fixed. An empty list implies that fixed values will be set based on lags.

Value

a list with starting values for the parameters of the selected model. Possible outputs are:

alpha	the intercept
beta	the coefficients for the regressors
phi	the AR coefficients
theta	for BARC models, the map parameter. For any other model, the MA coefficients
d	the long memory parameter
nu	the precison parameter

Examples

```
mu = 0.5
nu = 20

yt = rbeta(100, shape1 = mu*nu, shape2 = (1-mu)*nu)
coefs.start(model = "BARFIMA", yt = yt,
            linkg = "linear", d = FALSE,
            y.lower = 0, y.upper = 1)

yt = rgamma(100, shape = nu, rate = mu*nu)
coefs.start(model = "GARFIMA", yt = yt,
            linkg = "linear", d = FALSE,
            y.lower = 0, y.upper = Inf)
```

---

fit.control *Default control list*

Description

Sets default values for constants used by the optimization functions in FORTRAN

Usage

```
fit.control(control = list())
```

Arguments

`control`	a list with configurations to be passed to the optimization subroutines. Missing arguments will receive default values. See 'Details'.

Details

The `control` argument is a list that can supply any of the following components:

`method` The optimization method. Current available options are `"L-BFGS-B"` and `"Nelder-Mead"`. Default is `"L-BFGS-B"`.

`maxit` The maximum number of iterations. Defaults to `1000`.

`iprint` The frequency of reports if `control$trace` is positive. Defaults is -1 (no report).

- For `"L-BFGS-B"` method:
  iprint<0 no output is generated;
  iprint=0 print only one line at the last iteration;
  0<iprint<99 print also f and |proj g| every iprint iterations;
  iprint=99 print details of every iteration except n-vectors;
  iprint=100 print also the changes of active set and final x;
  iprint>100 print details of every iteration including x and g;
- For `"Nelder-Mead"` method:
  iprint<0 No printing
  iprint=0 Printing of parameter values and the function Value after initial evidence of convergence.
  iprint>0 As for iprint = 0 plus progress reports after every Iprint evaluations, plus printing for the initial simplex.

`factr` controls the convergence of the `"L-BFGS-B"` method. Convergence occurs when the reduction in the objective is within this factor of the machine tolerance. The iteration will stop when

$$(f^k - f^{k+1})/max\{|f^k|, |f^{k+1}|, 1\} \leq factr * epsmch$$

where epsmch is the machine precision, which is automatically generated by the code. Typical values for `factr`: 1.e+12 for low accuracy; 1.e+7 for moderate accuracy; 1.e+1 for extremely high accuracy. Default is `1e7`, that is a tolerance of about `1e-8`.

`pgtol` helps control the convergence of the `"L-BFGS-B"` method. It is a tolerance on the projected gradient in the current search direction. the iteration will stop when

$$max\{|projg_i|, i = 1, ..., n\} \leq pgtol$$

where pg_i is the ith component of the projected gradient. Default is `1e-12`.

`stopcr` The criterion applied to the standard deviation of the values of objective function at the points of the simplex, for "Nelder-Mead" method.

Value

a list with all arguments in 'Details'.

Examples

```
BTSR::fit.control()
```

GARFIMA.functions *Functions to simulate, extract components and fit GARFIMA models*

Description

These functions can be used to simulate, extract components and fit any model of the class garfima. A model with class garfima is a special case of a model with class btsr . See 'The BTSR structure' in btsr.functions for more details on the general structure.

The γARMA model, the gamma regression and a i.i.d. sample from a gamma distribution can be obtained as special cases. See 'Details'.

Usage

```
GARFIMA.sim(n = 1, burn = 0, xreg = NULL, coefs = list(alpha = 0, beta
  = NULL, phi = NULL, theta = NULL, d = 0, nu = 20), y.start = NULL,
  xreg.start = NULL, xregar = TRUE, error.scale = 0, complete = FALSE,
  inf = 1000, linkg = c("log", "log"), seed = NULL, rngtype = 2,
  debug = FALSE)

GARFIMA.extract(yt, xreg = NULL, nnew = 0, xnew = NULL, p, q,
  coefs = list(), lags = list(), fixed.values = list(),
  fixed.lags = list(), y.start = NULL, xreg.start = NULL,
  xregar = TRUE, error.scale = 0, inf = 1000, m = 0, linkg = c("log",
  "log"), llk = TRUE, sco = FALSE, info = FALSE, extra = FALSE,
  debug = FALSE)

GARFIMA.fit(yt, xreg = NULL, nnew = 0, xnew = NULL, p = 0, d = TRUE,
  q = 0, m = 0, inf = 1000, start = list(), ignore.start = FALSE,
  lags = list(), fixed.values = list(), fixed.lags = list(),
  lower = list(nu = 0), upper = list(nu = Inf), linkg = c("log", "log"),
  sco = TRUE, info = FALSE, extra = FALSE, xregar = TRUE,
  y.start = NULL, xreg.start = NULL, error.scale = 0, control = list(),
  report = TRUE, debug = FALSE, ...)
```

Arguments

n	a strictly positive integer. The sample size of yt (after burn-in). Default is 1.
burn	a non-negative integer. The length of the "burn-in" period. Default is 0.
xreg	optionally, a vector or matrix of external regressors. For simulation purposes, the length of xreg must be n+burn. Default is NULL. For extraction or fitting purposes, the length of xreg must be the same as the length of the observed time series y_t.

`coefs` a list with the coefficients of the model. An empty list will result in an error. The arguments that can be passed through this list are:

- `alpha` optionally, a numeric value corresponding to the intercept. If the argument is missing, it will be treated as zero. See 'The BTSR structure' in `btsr.functions`.
- `beta` optionally, a vector of coefficients corresponding to the regressors in `xreg`. If `xreg` is provided but `beta` is missing in the `coefs` list, an error message is issued.
- `phi` optionally, for the simulation function this must be a vector of size p, corresponding to the autoregressive coefficients (including the ones that are zero), where p is the AR order. For the extraction and fitting functions, this is a vector with the non-fixed values in the vector of autoregressive coefficients.
- `theta` optionally, for the simulation function this must be a vector of size q, corresponding to the moving average coefficients (including the ones that are zero), where q is the MA order. For the extraction and fitting functions, this is a vector with the non-fixed values in the vector of moving average coefficients.
- `d` optionally, a numeric value corresponding to the long memory parameter. If the argument is missing, it will be treated as zero.
- `nu` the dispersion parameter. If missing, an error message is issued.

`y.start` optionally, an initial value for yt (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $g_2(y_t) = 0$, for $t < 1$.

`xreg.start` optionally, a vector of initial value for xreg (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $X_t = 0$, for $t < 1$. If `xregar = FALSE` this argument is ignored.

`xregar` logical; indicates if xreg is to be included in the AR part of the model. See 'The BTSR structure'. Default is `TRUE`.

`error.scale` the scale for the error term. See 'The BTSR structure' in `btsr.functions`. Default is 0.

`complete` logical; if `FALSE` the function returns only the simulated time series yt, otherwise, additional time series are provided. Default is `FALSE`

`inf` the truncation point for infinite sums. Default is 1,000. In practice, the Fortran subroutine uses $inf = q$, if $d = 0$.

`linkg` character or a two character vector indicating which links must be used in the model. See 'The BTSR structure' in `btsr.functions` for details and `link.btsr` for valid links. If only one value is provided, the same link is used for mu_t and for y_t in the AR part of the model. Default is `c("log", "log")`. For the linear link, the constant will be always 1.

`seed` optionally, an integer which gives the value of the fixed seed to be used by the random number generator. If missing, a random integer is chosen uniformly from 1,000 to 10,000.

`rngtype` optionally, an integer indicating which random number generator is to be used. Default is 2: the Mersenne Twister algorithm. See 'Common Arguments' in `btsr.functions`.

debug	logical, if TRUE the output from FORTRAN is return (for debugging purposes). Default is FALSE for all models.
yt	a numeric vector with the observed time series. If missing, an error message is issued.
nnew	optionally, the number of out-of sample predicted values required. Default is 0.
xnew	a vector or matrix, with nnew observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. If xreg = NULL, xnew is ignored.
p	a non-negative integer. The order of AR polynomial. If missing, the value of p is calculated from length(coefs$phi) and length(fixed.values$phi). For fitting, the default is 0.
q	a non-negative integer. The order of the MA polynomial. If missing, the value of q is calculated from length(coefs$theta) and length(fixed.values$theta). For fitting, the default is 0.
lags	optionally, a list with the lags that the values in coefs correspond to. The names of the entries in this list must match the ones in coefs. For one dimensional coefficients, the lag is obviously always 1 and can be suppressed. An empty list indicates that either the argument fixed.lags is provided or all lags must be used.
fixed.values	optionally, a list with the values of the coefficients that are fixed. By default, if a given vector (such as the vector of AR coefficients) has fixed values and the corresponding entry in this list is empty, the fixed values are set as zero. The names of the entries in this list must match the ones in coefs.
fixed.lags	optionally, a list with the lags that the fixed values in fixed.values correspond to. The names of the entries in this list must match the ones in fixed.values. ##' For one dimensional coefficients, the lag is obviously always 1 and can be suppressed. If an empty list is provided and the model has fixed lags, the argument lags is used as reference.
m	a non-negative integer indicating the starting time for the sum of the partial log-likelihoods, that is $\ell = \sum_{t=m+1}^{n} \ell_t$. Default is 0.
llk	logical, if TRUE the value of the log-likelihood function is returned. Default is TRUE.
sco	logical, if TRUE the score vector is returned. Default is FALSE.
info	logical, if TRUE the information matrix is returned. Default is FALSE. For the fitting function, info is automatically set to TRUE when report = TRUE.
extra	logical, if TRUE the matrices and vectors used to calculate the score vector and the information matrix are returned. Default is FALSE.
d	logical, if TRUE, the parameter d is included in the model either as fixed or non-fixed. If d = FALSE the value is fixed as 0. The default is TRUE.
start	a list with the starting values for the non-fixed coefficients of the model. If an empty list is provided, the function coefs.start is used to obtain starting values for the parameters.

ignore.start	logical, if starting values are not provided, the function uses the default values and ignore.start is ignored. In case starting values are provided and ignore.start = TRUE, those starting values are ignored and recalculated. The default is FALSE.
lower	optionally, list with the lower bounds for the parameters. The names of the entries in these lists must match the ones in start. The default is to assume that the parameters have no lower bound except for nu, for which de default is 0. Only the bounds for bounded parameters need to be specified.
upper	optionally, list with the upper bounds for the parameters. The names of the entries in these lists must match the ones in start. The default is to assume that the parameters have no upper bound. Only the bounds for bounded parameters need to be specified.
control	a list with configurations to be passed to the optimization subroutines. Missing arguments will receive default values. See *fit.control*.
report	logical, if TRUE the summary from model estimation is printed and info is automatically set to TRUE. Default is TRUE.
...	further arguments passed to the internal functions.

Details

The γARMA model and the gamma regression can be obtained as special cases of the γARFIMA model.

- γARFIMA: is obtained by default.
- γARMA: is obtained by setting d = 0.
- gamma regression: is obtained by setting p = 0, q = 0 and d = FALSE. The error.scale is irrelevant. The second argument in linkg is irrelevant.
- an i.i.d. sample from a Gamma distribution with parameters shape and scale (compatible with the one from rgamma) is obtained by setting linkg = "linear", p = 0, q = 0, coefs$d = 0, d = FALSE and, in the coefficient list, alpha = shape*scale and nu = shape. (error.scale and xregar are irrelevant)

The function GARFIMA.sim generates a random sample from a γARFIMA(p,d,q) model.

The function GARFIMA.extract allows the user to extract the components y_t, μ_t, $\eta_t = g(\mu_t)$, r_t, the log-likelihood, and the vectors and matrices used to calculate the score vector and the information matrix associated to a given set of parameters.

This function can be used by any user to create an objective function that can be passed to optimization algorithms not available in the BTSR Package.

The function GARFIMA.fit fits a GARFIMA model to a given univariate time series. For now, available optimization algorithms are "L-BFGS-B" and "Nelder-Mead". Both methods accept bounds for the parameters. For "Nelder-Mead", bounds are set via parameter transformation.

Value

The function GARFIMA.sim returns the simulated time series yt by default. If complete = TRUE, a list with the following components is returned instead:

- model: string with the text "GARFIMA"
- yt: the simulated time series
- mut: the conditional mean
- etat: the linear predictor $g(\mu_t)$
- error: the error term r_t
- xreg: the regressors (if included in the model).
- debug: the output from FORTRAN (if requested).

The function GARFIMA.extract returns a list with the following components.

- model: string with the text "GARFIMA"
- coefs: the coefficients of the model passed through the coefs argument
- yt: the observed time series
- gyt: the transformed time series $g_2(y_t)$
- mut: the conditional mean
- etat: the linear predictor $g_1(\mu_t)$
- error: the error term r_t
- xreg: the regressors (if included in the model).
- sll: the sum of the conditional log-likelihood (if requested)
- sco: the score vector (if requested)
- info: the information matrix (if requested)
- Drho, T, E, h: additional matrices and vectors used to calculate the score vector and the information matrix. (if requested)
- yt.new: the out-of-sample forecast (if requested)
- out.Fortran: FORTRAN output (if requested)

The function btsr.fit returns a list with the following components. Each particular model can have additional components in this list.

- model: string with the text "GARFIMA"
- convergence: An integer code. 0 indicates successful completion. The error codes depend on the algorithm used.
- message: A character string giving any additional information returned by the optimizer, or NULL.
- counts: an integer giving the number of function evaluations.
- control: a list of control parameters.
- start: the starting values used by the algorithm.
- coefficients: The best set of parameters found.
- n: the sample size used for estimation.
- series: the observed time series
- gyt: the transformed time series $g_2(y_t)$

- `fitted.values`: the conditional mean, which corresponds to the in-sample forecast, also denoted fitted values
- `etat`: the linear predictor $g_1(\mu_t)$
- `error.scale`: the scale for the error term.
- `error`: the error term r_t
- `residual`: the observed minus the fitted values. The same as the `error` term if `error.scale` = `0`.
- `forecast`: the out-of-sample forecast (if requested).
- `xnew`: the observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. Only inlcudes if `xreg` is not `NULL` and `nnew` > `0`.
- `sll`: the sum of the conditional log-likelihood (if requested)
- `info.Matrix`: the information matrix (if requested)
- `configs`: a list with the configurations adopted to fit the model. This information is used by the prediction function.
- `out.Fortran`: FORTRAN output (if requested)
- `call`: a string with the description of the fitted model.

See Also

`btsr.sim`

`btsr.extract`

`btsr.fit`

Examples

```
# Generating a Gamma model were mut does not vary with time
# yt ~ Gamma(a,b), a = nu (shape), b = mu/nu (scale)

y <- GARFIMA.sim(linkg = "linear", n = 1000, seed = 2021,
                 coefs = list(alpha = 0.2, nu = 20))
hist(y)

 #-------------------------------------------------------------
 # Generating a Gamma model were mut does not vary with time
 # yt ~ Gamma(a,b), a = nu (shape), b = mu/nu (scale)
 #-------------------------------------------------------------

 m1 <- GARFIMA.sim(linkg = "linear",n = 100,
                   complete = TRUE, seed = 2021,
                   coefs = list(alpha = 0.2, nu = 20))

 #-------------------------------------------------------------
 #  Extracting the conditional time series given yt and
 #  a set of parameters
 #-------------------------------------------------------------

 # Assuming that all coefficients are non-fixed
```

```
e1 = GARFIMA.extract(yt = m1$yt, coefs = list(alpha = 0.2, nu = 20),
                     link = "linear", llk = TRUE,
                     sco = TRUE, info = TRUE)

#-----------------------------------------------------
# comparing the simulated and the extracted values
#-----------------------------------------------------
cbind(head(m1$mut), head(e1$mut))

#-----------------------------------------------------------
# the log-likelihood, score vector and information matrix
#-----------------------------------------------------------
e1$sll
e1$score
e1$info.Matrix


# Generating a Beta model were mut does not vary with time
# yt ~ Beta(a,b), a = mu*nu, b = (1-mu)*nu

y <- GARFIMA.sim(linkg = "linear", n = 100, seed = 2021,
                 coefs = list(alpha = 0.2, nu = 20))

# fitting the model
f <- GARFIMA.fit(yt = y, report = TRUE,
                  start = list(alpha = 0.5, nu = 10),
                  linkg = "linear", d = FALSE)
```

KARFIMA.functions *Functions to simulate, extract components and fit KARFIMA models*

Description

These functions can be used to simulate, extract components and fit any model of the class karfima. A model with class karfima is a special case of a model with class btsr . See 'The BTSR structure' in btsr.functions for more details on the general structure.

The KARMA model, the Kumaraswamy regression and a i.i.d. sample from a Kumaraswamy distribution can be obtained as special cases. See 'Details'.

Usage

```
KARFIMA.sim(n = 1, burn = 0, xreg = NULL, rho = 0.5, y.lower = 0,
  y.upper = 1, coefs = list(alpha = 0, beta = NULL, phi = NULL, theta =
  NULL, d = 0, nu = 20), y.start = NULL, xreg.start = NULL,
  xregar = TRUE, error.scale = 1, complete = FALSE, inf = 1000,
  linkg = c("logit", "logit"), seed = NULL, rngtype = 2, debug = FALSE)

KARFIMA.extract(yt, xreg = NULL, nnew = 0, xnew = NULL, p, q,
```

```
  rho = 0.5, y.lower = 0, y.upper = 1, coefs = list(), lags = list(),
  fixed.values = list(), fixed.lags = list(), y.start = NULL,
  xreg.start = NULL, xregar = TRUE, error.scale = 1, inf = 1000,
  m = 0, linkg = c("logit", "logit"), llk = TRUE, sco = FALSE,
  info = FALSE, extra = FALSE, debug = FALSE)

KARFIMA.fit(yt, xreg = NULL, nnew = 0, xnew = NULL, p = 0, d = TRUE,
  q = 0, m = 0, inf = 1000, rho = 0.5, y.lower = 0, y.upper = 1,
  start = list(), ignore.start = FALSE, lags = list(),
  fixed.values = list(), fixed.lags = list(), lower = list(nu = 0),
  upper = list(nu = Inf), linkg = c("logit", "logit"), sco = FALSE,
  info = FALSE, extra = FALSE, xregar = TRUE, y.start = NULL,
  xreg.start = NULL, error.scale = 1, control = list(), report = TRUE,
  debug = FALSE, ...)
```

Arguments

n	a strictly positive integer. The sample size of yt (after burn-in). Default is 1.
burn	a non-negative integer. The length of the "burn-in" period. Default is 0.
xreg	optionally, a vector or matrix of external regressors. For simulation purposes, the length of xreg must be n+burn. Default is NULL. For extraction or fitting purposes, the length of xreg must be the same as the length of the observed time series y_t.
rho	a positive number, between 0 and 1, indicating the quantile to be modeled so that μ_t is the conditional rho-quantile.
y.lower	the lower limit for the density support. Default is 0.
y.upper	the upper limit for the density support. Default is 1.
coefs	a list with the coefficients of the model. An empty list will result in an error. The arguments that can be passed through this list are:

- alpha optionally, a numeric value corresponding to the intercept. If the argument is missing, it will be treated as zero. See 'The BTSR structure' in btsr.functions.
- beta optionally, a vector of coefficients corresponding to the regressors in xreg. If xreg is provided but beta is missing in the coefs list, an error message is issued.
- phi optionally, for the simulation function this must be a vector of size p, corresponding to the autoregressive coefficients (including the ones that are zero), where p is the AR order. For the extraction and fitting functions, this is a vector with the non-fixed values in the vector of autoregressive coefficients.
- theta optionally, for the simulation function this must be a vector of size q, corresponding to the moving average coefficients (including the ones that are zero), where q is the MA order. For the extraction and fitting functions, this is a vector with the non-fixed values in the vector of moving average coefficients.
- d optionally, a numeric value corresponding to the long memory parameter. If the argument is missing, it will be treated as zero.

- nu the dispersion parameter. If missing, an error message is issued.

`y.start` optionally, an initial value for yt (to be used in the recursions). Default is NULL, in which case, the recursion assumes that $g_2(y_t) = 0$, for $t < 1$.

`xreg.start` optionally, a vector of initial value for xreg (to be used in the recursions). Default is NULL, in which case, the recursion assumes that $X_t = 0$, for $t < 1$. If `xregar` = FALSE this argument is ignored.

`xregar` logical; indicates if xreg is to be included in the AR part of the model. See 'The BTSR structure'. Default is TRUE.

`error.scale` the scale for the error term. See 'The BTSR structure' in `btsr.functions`. Default is 1.

`complete` logical; if FALSE the function returns only the simulated time series yt, otherwise, additional time series are provided. Default is FALSE

`inf` the truncation point for infinite sums. Default is 1,000. In practice, the Fortran subroutine uses $inf = q$, if $d = 0$.

`linkg` character or a two character vector indicating which links must be used in the model. See 'The BTSR structure' in `btsr.functions` for details and `link.btsr` for valid links. If only one value is provided, the same link is used for mu_t and for y_t in the AR part of the model. Default is `c("logit", "logit")`. For the linear link, the constant will be always 1.

`seed` optionally, an integer which gives the value of the fixed seed to be used by the random number generator. If missing, a random integer is chosen uniformly from 1,000 to 10,000.

`rngtype` optionally, an integer indicating which random number generator is to be used. Default is 2: the Mersenne Twister algorithm. See 'Common Arguments' in `btsr.functions`.

`debug` logical, if TRUE the output from FORTRAN is return (for debugging purposes). Default is FALSE for all models.

`yt` a numeric vector with the observed time series. If missing, an error message is issued.

`nnew` optionally, the number of out-of sample predicted values required. Default is 0.

`xnew` a vector or matrix, with nnew observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. If `xreg` = NULL, `xnew` is ignored.

`p` a non-negative integer. The order of AR polynomial. If missing, the value of p is calculated from length(coefs$phi) and length(fixed.values$phi). For fitting, the default is 0.

`q` a non-negative integer. The order of the MA polynomial. If missing, the value of q is calculated from length(coefs$theta) and length(fixed.values$theta). For fitting, the default is 0.

`lags` optionally, a list with the lags that the values in `coefs` correspond to. The names of the entries in this list must match the ones in `coefs`. For one dimensional coefficients, the `lag` is obviously always 1 and can be suppressed. An empty list indicates that either the argument `fixed.lags` is provided or all lags must be used.

fixed.values	optionally, a list with the values of the coefficients that are fixed. By default, if a given vector (such as the vector of AR coefficients) has fixed values and the corresponding entry in this list is empty, the fixed values are set as zero. The names of the entries in this list must match the ones in coefs.
fixed.lags	optionally, a list with the lags that the fixed values in fixed.values correspond to. The names of the entries in this list must match the ones in fixed.values. ##' For one dimensional coefficients, the lag is obviously always 1 and can be suppressed. If an empty list is provided and the model has fixed lags, the argument lags is used as reference.
m	a non-negative integer indicating the starting time for the sum of the partial log-likelihoods, that is $\ell = \sum_{t=m+1}^{n} \ell_t$. Default is 0.
llk	logical, if TRUE the value of the log-likelihood function is returned. Default is TRUE.
sco	logical, if TRUE the score vector is returned. Default is FALSE.
info	logical, if TRUE the information matrix is returned. Default is FALSE. For the fitting function, info is automatically set to TRUE when report = TRUE.
extra	logical, if TRUE the matrices and vectors used to calculate the score vector and the information matrix are returned. Default is FALSE.
d	logical, if TRUE, the parameter d is included in the model either as fixed or non-fixed. If d = FALSE the value is fixed as 0. The default is TRUE.
start	a list with the starting values for the non-fixed coefficients of the model. If an empty list is provided, the function coefs.start is used to obtain starting values for the parameters.
ignore.start	logical, if starting values are not provided, the function uses the default values and ignore.start is ignored. In case starting values are provided and ignore.start = TRUE, those starting values are ignored and recalculated. The default is FALSE.
lower	optionally, list with the lower bounds for the parameters. The names of the entries in these lists must match the ones in start. The default is to assume that the parameters have no lower bound except for nu, for which de default is 0. Only the bounds for bounded parameters need to be specified.
upper	optionally, list with the upper bounds for the parameters. The names of the entries in these lists must match the ones in start. The default is to assume that the parameters have no upper bound. Only the bounds for bounded parameters need to be specified.
control	a list with configurations to be passed to the optimization subroutines. Missing arguments will receive default values. See *fit.control*.
report	logical, if TRUE the summary from model estimation is printed and info is automatically set to TRUE. Default is TRUE.
...	further arguments passed to the internal functions.

Details

The KARMA model and the Kumaraswamy regression can be obtained as special cases of the KARFIMA model.

- KARFIMA: is obtained by default.
- KARMA: is obtained by setting d = 0.
- Kumaraswamy regression: is obtained by setting p = 0, q = 0 and d = FALSE. The error.scale is irrelevant. The second argument in linkg is irrelevant.
- an i.i.d. sample from a Kumaraswamy distribution is obtained by setting linkg = "linear", p = 0, q = 0, coefs$d = 0, d = FALSE. (error.scale and xregar are irrelevant)

The function KARFIMA.sim generates a random sample from a KARFIMA(p,d,q) model.

The function KARFIMA.extract allows the user to extract the components y_t, μ_t, $\eta_t = g(\mu_t)$, r_t, the log-likelihood, and the vectors and matrices used to calculate the score vector and the information matrix associated to a given set of parameters.

This function can be used by any user to create an objective function that can be passed to optimization algorithms not available in the BTSR Package.

The function KARFIMA.fit fits a KARFIMA model to a given univariate time series. For now, available optimization algorithms are "L-BFGS-B" and "Nelder-Mead". Both methods accept bounds for the parameters. For "Nelder-Mead", bounds are set via parameter transformation.

Value

The function KARFIMA.sim returns the simulated time series yt by default. If complete = TRUE, a list with the following components is returned instead:

- model: string with the text "KARFIMA"
- yt: the simulated time series
- mut: the conditional mean
- etat: the linear predictor $g(\mu_t)$
- error: the error term r_t
- xreg: the regressors (if included in the model).
- debug: the output from FORTRAN (if requested).

The function KARFIMA.extract returns a list with the following components.

- model: string with the text "KARFIMA"
- coefs: the coefficients of the model passed through the coefs argument
- yt: the observed time series
- gyt: the transformed time series $g_2(y_t)$
- mut: the conditional mean
- etat: the linear predictor $g_1(\mu_t)$
- error: the error term r_t
- xreg: the regressors (if included in the model).
- sll: the sum of the conditional log-likelihood (if requested)
- sco: the score vector (if requested)
- info: the information matrix (if requested)

- Drho, T, E, h: additional matrices and vectors used to calculate the score vector and the information matrix. (if requested)
- yt.new: the out-of-sample forecast (if requested)
- out.Fortran: FORTRAN output (if requested)

The function btsr.fit returns a list with the following components. Each particular model can have additional components in this list.

- model: string with the text "KARFIMA"
- convergence: An integer code. 0 indicates successful completion. The error codes depend on the algorithm used.
- message: A character string giving any additional information returned by the optimizer, or NULL.
- counts: an integer giving the number of function evaluations.
- control: a list of control parameters.
- start: the starting values used by the algorithm.
- coefficients: The best set of parameters found.
- n: the sample size used for estimation.
- series: the observed time series
- gyt: the transformed time series $g_2(y_t)$
- fitted.values: the conditional mean, which corresponds to the in-sample forecast, also denoted fitted values
- etat: the linear predictor $g_1(\mu_t)$
- error.scale: the scale for the error term.
- error: the error term r_t
- residual: the observed minus the fitted values. The same as the error term if error.scale = 0.
- forecast: the out-of-sample forecast (if requested).
- xnew: the observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. Only inlcudes if xreg is not NULL and nnew > 0.
- sll: the sum of the conditional log-likelihood (if requested)
- info.Matrix: the information matrix (if requested)
- configs: a list with the configurations adopted to fit the model. This information is used by the prediction function.
- out.Fortran: FORTRAN output (if requested)
- call: a string with the description of the fitted model.

See Also

btsr.sim

btsr.extract

btsr.fit

Examples

```
# Generating a Kumaraswamy model were mut does not vary with time
# For linear link, alpha = mu
#
# Warning:
#      |log(1-rho)| >> |log(1 - mu^nu)|
# may cause numerical instability.

y <- KARFIMA.sim(linkg = "linear", n = 1000, seed = 2021,
                 coefs = list(alpha = 0.7, nu = 2))
hist(y)

 #-------------------------------------------------------------
 # Generating a Kumaraswamy model were mut does not vary with time
 # For linear link, alpha = mu
 #
 # Warning:
 #      |log(1-rho)| >> |log(1 - mu^nu)|
 # may cause numerical instability.
 #-------------------------------------------------------------

 m1 <- KARFIMA.sim(linkg = "linear",n = 100,
                   complete = TRUE, seed = 2021,
                   coefs = list(alpha = 0.7, nu = 2))

 #-------------------------------------------------------------
 #  Extracting the conditional time series given yt and
 #  a set of parameters
 #-------------------------------------------------------------

 # Assuming that all coefficients are non-fixed
 e1 = KARFIMA.extract(yt = m1$yt, coefs = list(alpha = 0.7, nu = 2),
                      link = "linear", llk = TRUE,
                      sco = TRUE, info = TRUE)

 #-----------------------------------------------------
 # comparing the simulated and the extracted values
 #-----------------------------------------------------
 cbind(head(m1$mut), head(e1$mut))

 #-----------------------------------------------------------
 # the log-likelihood, score vector and information matrix
 #-----------------------------------------------------------
 e1$sll
 e1$score
 e1$info.Matrix


# Generating a Kumaraswamy model were mut does not vary with time
# For linear link, alpha = mu
#
# Warning:
```

```
#      |log(1-rho)| >> |log(1 - mu^nu)|
# may cause numerical instability.

y <- KARFIMA.sim(linkg = "logit", n = 100, seed = 2021,
                coefs = list(alpha = 0.7, nu = 2))

# fitting the model
f <- KARFIMA.fit(yt = y, report = TRUE,
                 start = list(alpha = 0.5, nu = 1),
                 linkg = "logit", d = FALSE)
```

`link.btsr` *Create a Link for BTSR models*

Description

Given the name of a link, this function returns a link function, an inverse link function, the derivative $d\eta/d\mu$ and the derivative $d\mu/d\eta$.

Usage

```
link.btsr(link)
```

Arguments

`link`	character; one of `"linear"`, `"logit"`, `"log"`, `"loglog"`, `"cloglog"`. See 'Details'.

Details

The available links are:

linear: $f(x) = ax$, for a real. The parameter is set using the argument `ctt.ll`, when invoking the functions created by `link.btsr`

logit: $f(x) = log(x/(1-x))$

log: $f(x) = log(x)$

loglog: $f(x) = log(-log(x))$

cloglog: $f(x) = log(-log(1-x))$

Value

An object of class `"link-btsr"`, a list with components

`linkfun`	Link function `function(mu)`
`linkinv`	Inverse link function `function(eta)`
`linkdif`	Derivative `function(mu)` $d\eta/d\mu$
`mu.eta`	Derivative `function(eta)` $d\mu/d\eta$
`name`	a name to be used for the link

Examples

```
mylink <- BTSR::link.btsr("linear")
y = 0.8
a = 3.4
gy = a*y

mylink$linkfun(mu = y, ctt.ll = a); gy
mylink$linkinv(eta = gy, ctt.ll = a); y
mylink$diflink(mu = y, ctt.ll = a); a
mylink$mu.eta(eta = gy, ctt.ll = a); 1/a
```

`predict.btsr` *Predict method for BTSR*

Description

Predicted values based on btsr object.

Usage

```
## S3 method for class 'btsr'
predict(object, newdata, nnew = 0, ...)
```

Arguments

`object`	Object of class inheriting from `"btsr"`
`newdata`	A matrix with new values for the regressors. If omitted and `"xreg"` is present in the model, the fitted values are returned. If the model does not include regressors, the functions will use the value of `nnew`.
`nnew`	number of out-of-sample forecasts required. If `newdata` is provided, `nnew` is ignored.
`...`	further arguments passed to or from other methods.

Details

`predict.btsr` produces predicted values, obtained by evaluating the regression function in the frame `newdata`.

If `newdata` is omitted the predictions are based on the data used for the fit.

For now, prediction intervals are not provided.

Value

A list with the following arguments

series	The original time series yt.
xreg	The original regressors (if any).
fitted.values	The in-sample forecast given by μ_t.
etat	In-sample values of $g(\mu[t])$.
error	The error term (depends on the argument error.scale)
residuals	The (in-sample) residuals, that is, the observed minus the predicted values. Same as error when error.scale = 0
forecast	The predicted values for yt.
TS	only for "BARC" models. The iterated map.
Ts.forecast	only for "BARC" models. The predicted values of the iterated map.

Examples

```
 #-------------------------------------------------------------
 # Generating a Beta model were mut does not vary with time
 # yt ~ Beta(a,b), a = mu*nu, b = (1-mu)*nu
 #-------------------------------------------------------------

y <- btsr.sim(model= "BARFIMA", linkg = "linear",
               n = 100, seed = 2021,
               coefs = list(alpha = 0.2, nu = 20))

# fitting the model
f <- btsr.fit(model = "BARFIMA", yt = y, report = TRUE,
              start = list(alpha = 0.5, nu = 10),
              linkg = "linear", d = FALSE)

pred = predict(f, nnew = 5)
pred$forecast
```

---

print.btsr *Print Method of class BTSR*

Description

Print method for objects of class btsr.

Usage

```
## S3 method for class 'btsr'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	object of class btsr.
digits	minimal number of significant digits, see print.default.
...	further arguments to be passed to or from other methods. They are ignored in this function

Details

Users are not encouraged to call these internal functions directly. Internal functions for package BTSR.

Value

Invisibly returns its argument, x.

---

summary *Summary Method of class BTSR*

Description

summary method for class "btsr".

Usage

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

## S3 method for class 'summary.btsr'
print(x, digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	object of class "btsr".
...	further arguments passed to or from other methods.
x	an object of class "summary.btsr", usually, a result of a call to summary.btsr.
digits	minimal number of significant digits, see print.default.
signif.stars	logical. If TRUE, 'significance stars' are printed for each coefficient.

Details

print.summary.btsr tries to be smart about formatting the coefficients, standard errors, etc. and additionally provides 'significance stars'.

Value

The function `summary.btsr` computes and returns a list of summary statistics of the fitted model given in `object`. Returns a list of class `summary.btsr`, which contains the following components:

`model`	the corresponding model.
`call`	the matched call.
`residuals`	the residuals of the model. Depends on the definition of `error.scale`. If error.scale= 1, $residuals = g(y) - g(\mu)$. If error.scale = 0, $residuals = y - \mu$.
`coefficients`	a $k \times 4$ matrix with columns for the estimated coefficient, its standard error, z-statistic and corresponding (two-sided) p-value. Aliased coefficients are omitted.
`aliased`	named logical vector showing if the original coefficients are aliased.
`sigma.res`	the square root of the estimated variance of the random error $$\hat{\sigma}^2 = \frac{1}{n-k}\sum_i r_i^2,$$ where r_i is the i-th residual, `residuals[i]`.
`df`	degrees of freedom, a 3-vector $(k, n-k, k*)$, the first being the number of non-aliased coefficients, the last being the total number of coefficients.
`vcov`	a $k \times k$ matrix of (unscaled) covariances. The inverse ov the information matrix.
`loglik`	the sum of the log-likelihood values
`aic`	the AIC value. $AIC = -2 * loglik + 2 * k$.
`bic`	the BIC value. $BIC = -2 * loglik + log(n) * k$.
`hqc`	the HQC value. $HQC = -2 * loglik + log(log(n)) * k$.

`UWARFIMA.functions` *Functions to simulate, extract components and fit UWARFIMA models*

Description

These functions can be used to simulate, extract components and fit any model of the class `uwarfima`. A model with class `uwarfima` is a special case of a model with class `btsr` . See 'The BTSR structure' in `btsr.functions` for more details on the general structure.

The UWARMA model, the Unit-Weibull regression and a i.i.d. sample from a Unit-Weibull distribution can be obtained as special cases. See 'Details'.

Usage

```
UWARFIMA.sim(n = 1, burn = 0, xreg = NULL, rho = 0.5,
  coefs = list(alpha = 0, beta = NULL, phi = NULL, theta = NULL, d = 0, nu =
  20), y.start = NULL, xreg.start = NULL, xregar = TRUE,
  error.scale = 1, complete = FALSE, inf = 1000, linkg = c("logit",
  "logit"), seed = NULL, rngtype = 2, debug = FALSE)

UWARFIMA.extract(yt, xreg = NULL, nnew = 0, xnew = NULL, p, q,
  rho = 0.5, coefs = list(), lags = list(), fixed.values = list(),
  fixed.lags = list(), y.start = NULL, xreg.start = NULL,
  xregar = TRUE, error.scale = 1, inf = 1000, m = 0,
  linkg = c("logit", "logit"), llk = TRUE, sco = FALSE, info = FALSE,
  extra = FALSE, debug = FALSE)

UWARFIMA.fit(yt, xreg = NULL, nnew = 0, xnew = NULL, p = 0, d = TRUE,
  q = 0, m = 0, inf = 1000, rho = 0.5, start = list(),
  ignore.start = FALSE, lags = list(), fixed.values = list(),
  fixed.lags = list(), lower = list(nu = 0), upper = list(nu = Inf),
  linkg = c("logit", "logit"), sco = FALSE, info = FALSE,
  extra = FALSE, xregar = TRUE, y.start = NULL, xreg.start = NULL,
  error.scale = 1, control = list(), report = TRUE, debug = FALSE, ...)
```

Arguments

n	a strictly positive integer. The sample size of yt (after burn-in). Default is 1.
burn	a non-negative integer. Length of the "burn-in" period. Default is 0.
xreg	optionally, a vector or matrix of external regressors. For simulation purposes, the length of xreg must be n+burn. Default is NULL. For extraction or fitting purposes, the length of xreg must be the same as the length of the observed time series y_t.
rho	a positive number, between 0 and 1, indicating the quantile to be modeled. In this case, μ_t corresponds to the conditional rho-quantile of the distribution.
coefs	a list with the coefficients of the model. An empty list will result in an error. The arguments that can be passed through this list are: • alpha optionally, a numeric value corresponding to the intercept. If the argument is missing, it will be treated as zero. See 'The BTSR structure' in btsr.functions. • beta optionally, a vector of coefficients corresponding to the regressors in xreg. If xreg is provided but beta is missing in the coefs list, an error message is issued. • phi optionally, for the simulation function this must be a vector of size p, corresponding to the autoregressive coefficients (including the ones that are zero), where p is the AR order. For the extraction and fitting functions, this is a vector with the non-fixed values in the vector of autoregressive coefficients.

- `theta` optionally, for the simulation function this must be a vector of size q, corresponding to the moving average coefficients (including the ones that are zero), where q is the MA order. For the extraction and fitting functions, this is a vector with the non-fixed values in the vector of moving average coefficients.
- `d` optionally, a numeric value corresponding to the long memory parameter. If the argument is missing, it will be treated as zero.
- `nu` is a shape parameter. If missing, an error message is issued.

`y.start` optionally, an initial value for yt (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $g_2(y_t) = 0$, for $t < 1$.

`xreg.start` optionally, a vector of initial value for xreg (to be used in the recursions). Default is `NULL`, in which case, the recursion assumes that $X_t = 0$, for $t < 1$. If `xregar = FALSE` this argument is ignored.

`xregar` logical; indicates if xreg is to be included in the AR part of the model. See 'The BTSR structure'. Default is `TRUE`.

`error.scale` the scale for the error term. See 'The BTSR structure' in `btsr.functions`. Default is 1.

`complete` logical; if `FALSE` the function returns only the simulated time series yt, otherwise, additional time series are provided (see below). Default is `FALSE`

`inf` the truncation point for infinite sums. Default is 1,000. In practice, the Fortran subroutine uses $inf = q$, if $d = 0$.

`linkg` character or a two character vector indicating which links must be used in the model. See 'The BTSR structure' in `btsr.functions` for details and `link.btsr` for valid links. If only one value is provided, the same link is used for mu_t and for y_t in the AR part of the model. Default is `c("logit", "logit")`. For the linear link, the constant will be always 1.

`seed` optionally, an integer which gives the value of the fixed seed to be used by the random number generator. If missing, a random integer is chosen uniformly from 1,000 to 10,000.

`rngtype` optionally, an integer indicating which random number generator is to be used. Default is 2: the Mersenne Twister algorithm. See 'Common Arguments' in `btsr.functions`.

`debug` logical, if `TRUE` the output from FORTRAN is return (for debugging purposes). Default is `FALSE` for all models.

`yt` a numeric vector with the observed time series. If missing, an error message is issued.

`nnew` optionally, the number of out-of sample predicted values required. Default is 0.

`xnew` a vector or matrix, with `nnew` observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. If `xreg = NULL`, `xnew` is ignored.

`p` a non-negative integer. The order of AR polynomial. If missing, the value of `p` is calculated from length(coefs$phi) and length(fixed.values$phi). For fitting, the default is 0.

q	a non-negative integer. The order of the MA polynomial. If missing, the value of q is calculated from length(coefs$theta) and length(fixed.values$theta). For fitting, the default is 0.
lags	optionally, a list with the lags that the values in coefs correspond to. The names of the entries in this list must match the ones in coefs. For one dimensional coefficients, the lag is obviously always 1 and can be suppressed. An empty list indicates that either the argument fixed.lags is provided or all lags must be used.
fixed.values	optionally, a list with the values of the coefficients that are fixed. By default, if a given vector (such as the vector of AR coefficients) has fixed values and the corresponding entry in this list is empty, the fixed values are set as zero. The names of the entries in this list must match the ones in coefs.
fixed.lags	optionally, a list with the lags that the fixed values in fixed.values correspond to. The names of the entries in this list must match the ones in fixed.values. ##' For one dimensional coefficients, the lag is obviously always 1 and can be suppressed. If an empty list is provided and the model has fixed lags, the argument lags is used as reference.
m	a non-negative integer indicating the starting time for the sum of the partial log-likelihoods, that is $\ell = \sum_{t=m+1}^{n} \ell_t$. Default is 0.
llk	logical, if TRUE the value of the log-likelihood function is returned. Default is TRUE.
sco	logical, if TRUE the score vector is returned. Default is FALSE.
info	logical, if TRUE the information matrix is returned. Default is FALSE. For the fitting function, info is automatically set to TRUE when report = TRUE.
extra	logical, if TRUE the matrices and vectors used to calculate the score vector and the information matrix are returned. Default is FALSE.
d	logical, if TRUE, the parameter d is included in the model either as fixed or non-fixed. If d = FALSE the value is fixed as 0. The default is TRUE.
start	a list with the starting values for the non-fixed coefficients of the model. If an empty list is provided, the function coefs.start is used to obtain starting values for the parameters.
ignore.start	logical, if starting values are not provided, the function uses the default values and ignore.start is ignored. In case starting values are provided and ignore.start = TRUE, those starting values are ignored and recalculated. The default is FALSE.
lower	optionally, list with the lower bounds for the parameters. The names of the entries in these lists must match the ones in start. The default is to assume that the parameters have no lower bound except for nu, for which de default is 0. Only the bounds for bounded parameters need to be specified.
upper	optionally, list with the upper bounds for the parameters. The names of the entries in these lists must match the ones in start. The default is to assume that the parameters have no upper bound. Only the bounds for bounded parameters need to be specified.
control	a list with configurations to be passed to the optimization subroutines. Missing arguments will receive default values. See *fit.control*.

report	logical, if TRUE the summary from model estimation is printed and info is automatically set to TRUE. Default is TRUE.
...	further arguments passed to the internal functions.

Details

The UWARMA model and the Unit-Weibull regression can be obtained as special cases of the UWARFIMA model.

- UWARFIMA: is obtained by default.
- UWARMA: is obtained by setting d = 0.
- Unit-Weibull regression: is obtained by setting p = 0, q = 0 and d = FALSE. The error.scale is irrelevant. The second argument in linkg is irrelevant.
- an i.i.d. sample from a Unit-Weibull distribution is obtained by setting linkg = "linear", p = 0, q = 0, coefs$d = 0, d = FALSE. (error.scale and xregar are irrelevant)

The function UWARFIMA.sim generates a random sample from a UWARFIMA(p,d,q) model.

The function UWARFIMA.extract allows the user to extract the components y_t, μ_t, $\eta_t = g(\mu_t)$, r_t, the log-likelihood, and the vectors and matrices used to calculate the score vector and the information matrix associated to a given set of parameters.

This function can be used by any user to create an objective function that can be passed to optimization algorithms not available in the BTSR Package.

The function UWARFIMA.fit fits a UWARFIMA model to a given univariate time series. For now, available optimization algorithms are "L-BFGS-B" and "Nelder-Mead". Both methods accept bounds for the parameters. For "Nelder-Mead", bounds are set via parameter transformation.

Value

The function UWARFIMA.sim returns the simulated time series yt by default. If complete = TRUE, a list with the following components is returned instead:

- model: string with the text "UWARFIMA"
- yt: the simulated time series
- mut: the conditional mean
- etat: the linear predictor $g(\mu_t)$
- error: the error term r_t
- xreg: the regressors (if included in the model).
- debug: the output from FORTRAN (if requested).

The function UWARFIMA.extract returns a list with the following components.

- model: string with the text "UWARFIMA"
- coefs: the coefficients of the model passed through the coefs argument
- yt: the observed time series
- gyt: the transformed time series $g_2(y_t)$

- mut: the conditional mean
- etat: the linear predictor $g_1(\mu_t)$
- error: the error term r_t
- xreg: the regressors (if included in the model).
- sll: the sum of the conditional log-likelihood (if requested)
- sco: the score vector (if requested)
- info: the information matrix (if requested)
- Drho, T, E, h: additional matrices and vectors used to calculate the score vector and the information matrix. (if requested)
- yt.new: the out-of-sample forecast (if requested)
- out.Fortran: FORTRAN output (if requested)

The function btsr.fit returns a list with the following components. Each particular model can have additional components in this list.

- model: string with the text "UWARFIMA"
- convergence: An integer code. 0 indicates successful completion. The error codes depend on the algorithm used.
- message: A character string giving any additional information returned by the optimizer, or NULL.
- counts: an integer giving the number of function evaluations.
- control: a list of control parameters.
- start: the starting values used by the algorithm.
- coefficients: The best set of parameters found.
- n: the sample size used for estimation.
- series: the observed time series
- gyt: the transformed time series $g_2(y_t)$
- fitted.values: the conditional mean, which corresponds to the in-sample forecast, also denoted fitted values
- etat: the linear predictor $g_1(\mu_t)$
- error.scale: the scale for the error term.
- error: the error term r_t
- residual: the observed minus the fitted values. The same as the error term if error.scale = 0.
- forecast: the out-of-sample forecast (if requested).
- xnew: the observations of the regressors observed/predicted values corresponding to the period of out-of-sample forecast. Only inlcudes if xreg is not NULL and nnew > 0.
- sll: the sum of the conditional log-likelihood (if requested)
- info.Matrix: the information matrix (if requested)
- configs: a list with the configurations adopted to fit the model. This information is used by the prediction function.
- out.Fortran: FORTRAN output (if requested)
- call: a string with the description of the fitted model.

See Also

btsr.sim

btsr.extract

btsr.fit

Examples

```
# Generating a Unit-Weibull  model were mut does not vary with time
# For linear link, alpha = mu

y <- UWARFIMA.sim(linkg = "linear", n = 1000, seed = 2021,
                  coefs = list(alpha = 0.7, nu = 2))
hist(y)

 #-------------------------------------------------------------
 # Generating a Unit-Weibull model were mut does not vary with time
 # For linear link, alpha = mu
 #-------------------------------------------------------------

 m1 <- UWARFIMA.sim(linkg = "linear",n = 100,
                    complete = TRUE, seed = 2021,
                    coefs = list(alpha = 0.7, nu = 2))

 #-------------------------------------------------------------
 #  Extracting the conditional time series given yt and
 #  a set of parameters
 #-------------------------------------------------------------

 # Assuming that all coefficients are non-fixed
 e1 = UWARFIMA.extract(yt = m1$yt, coefs = list(alpha = 0.7, nu = 2),
                       link = "linear", llk = TRUE,
                       sco = TRUE, info = TRUE)

 #-----------------------------------------------------
 # comparing the simulated and the extracted values
 #-----------------------------------------------------
 cbind(head(m1$mut), head(e1$mut))

 #-----------------------------------------------------------
 # the log-likelihood, score vector and information matrix
 #-----------------------------------------------------------
 e1$sll
 e1$score
 e1$info.Matrix


# Generating a Unit-Weibull model were mut does not vary with time
# For linear link, alpha = mu

y <- UWARFIMA.sim(linkg = "logit", n = 100, seed = 2021,
                  coefs = list(alpha = 0.7, nu = 2))
```

```
# fitting the model
f <- UWARFIMA.fit(yt = y, report = TRUE,
                  start = list(alpha = 0.5, nu = 1),
                  linkg = "logit", d = FALSE)
```

Index