Package 'carfima'

January 23, 2019

Type Package

Title Continuous-Time Fractionally Integrated ARMA Process for Irregularly Spaced Long-Memory Time Series Data

Version 2.0.1

Description We provide a toolbox to fit a continuous-time fractionally integrated ARMA process (CARFIMA) on univariate and irregularly spaced time series data via frequentist or Bayesian machinery. A general-order CARFIMA(p, H, q) model for p>q is specified in Tsai and Chan (2005) <doi:10.1111/j.1467-9868.2005.00522.x> and it involves (p+q+2) unknown model parameters, i.e., p AR parameters, q MA parameters, Hurst parameter H, and process uncertainty (standard deviation) sigma. The package produces their maximum likelihood estimates and asymptotic uncertainties using a global optimizer called the differential evolution algorithm. It also produces their posterior distributions via Metropolis within a Gibbs sampler equipped with adaptive Markov chain Monte Carlo for posterior sampling. These fitting procedures, however, may produce numerical errors if p>2. The toolbox also contains a function to simulate discrete time series data from CARFIMA(p, H, q) process given the model parameters and observation times.

License GPL-2

Encoding UTF-8

Imports Rcpp, DEoptim, Rdpack, MASS, cubature, numDeriv, stats, utils, truncnorm, invgamma

LinkingTo Rcpp, RcppArmadillo, cubature

RoxygenNote 6.1.1

RdMacros Rdpack

NeedsCompilation yes

Author Hyungsuk Tak [aut] (<https://orcid.org/0000-0003-0334-8742>), Henghsiu Tsai [aut], Kisung You [aut, cre] (<https://orcid.org/0000-0002-8584-459X>)

Maintainer Kisung You <kyou@nd.edu>

Repository CRAN

Date/Publication 2019-01-23 06:00:03 UTC

R topics documented:

`carfima` *Fitting a CARFIMA(p, H, q) model via frequentist or Bayesian machinery*

Description

A general-order CARFIMA(p, H, q) model for $p > q$ is

$$Y_t^{(p)} - \alpha_p Y_t^{(p-1)} - \cdots - \alpha_1 Y_t = \sigma(B_{t,H}^{(1)} + \beta_1 B_{t,H}^{(2)} + \cdots + \beta_q B_{t,H}^{(q+1)}),$$

where $B_{t,H} = B_t^H$ is the standard fractional Brownian motion, H is the Hurst parameter, and the superscript (j) indicates j-fold differentiation with respect to t; see Equation (1) of Tsai and Chan (2005) for details. The model has $p + q + 2$ unknown model parameters; p α_j's, q β_j's, H, and σ.

Usage

```
carfima(Y, time, ar.p, ma.q, method = c("mle", "bayes"), bayes.param.ini,
  bayes.param.scale, bayes.n.warm, bayes.n.sample)
```

Arguments

`Y`	A vector of length k for the observed data.
`time`	A vector of length k for the observation times.
`ar.p`	A positive integer for the order of the AR model. `ar.p` must be greater than `ma.q`. If `ar.p` is greater than 2, numerical errors may occur.
`ma.q`	A non-negative integer for the order of the MA model. `ma.q` must be smaller than `ar.p`.
`method`	Either "mle" or "bayes". Method "mle" conducts the MLE-based inference, producing MLEs and asymptotic uncertainties of the model parameters. Method "bayes" draws posterior samples of the model parameters.
`bayes.param.ini`	Only if `method` is "bayes". A vector of length $p + q + 2$ for the initial values of p α_j's, q β_j's, H, and σ to implement a Markov chain Monte Carlo method (Metropolis within Gibbs sampler). When a CARFIMA(2, H, 1) model is fitted, for example, users should set five initial values of α_1, α_2, β_1, H, and σ.

`bayes.param.scale`
: Only if `method` is "bayes". A vector of length $p + q + 2$ for jumping (proposal) scales of the Metropolis steps. Each number determines how far a Metropolis step reaches out in each parameter space. Note that the last number of this vector is the jumping scale to update σ^2 on a log scale. The adaptive MCMC automatically adjusts these jumping scales during the run.

`bayes.n.warm`
: Only if `method` is "bayes". A scalar for the number of burn-ins, i.e., the number of the first few iterations to be discarded to remove the effect of initial values.

`bayes.n.sample`
: Only if `method` is "bayes". A scalar for the number of posterior samples for each parameter.

Details

The function `carfima` fits the model, producing either their maximum likelihood estimates (MLEs) with their asymptotic uncertainties or their posterior samples according to its argument, `method`. The MLEs except σ are obtained from a profile likelihood by a global optimizer called the differential evolution algorithm on restricted ranges, i.e., (-0.99, -0.01) for each α_j, (-10, 10) for each β_j, and (0.51, 0.99) for H; the MLE of σ is then deterministically computed. The corresponding asymptotic uncertainties are based on a numerical hessian matrix calculation at the MLEs (see function `hessian` in **numDeriv**). It also computes the Akaike Information Criterion (AIC) that is -2(log likelihood $-p - q - 2$). The function `carfima` becomes numerically unstable if $p > 2$, and thus it may produce numerical errors. (The original Fortran code used in Tsai and Chan (2005) does not have this numerical issue because they use a different global optimizer called `DBCONF` from the IMSL Fortran library.)

The Bayesian approach uses independent prior distributions for the unknown model parameters; a Uniform(-0.99, -0.01) prior for each α_j, a Normal(0, 10^4) prior for each β_j, a Uniform(0.51, 0.99) prior for H for long memory process, and finally an inverse-Gamma(shape = 2.01, scale = 10^3) prior for σ^2. Posterior propriety holds because the prior distributions are jointly proper. It also adopts appropriate proposal density functions; a truncated Normal(current state, proposal scale) between -0.99 and -0.01 for each α_j, a Normal(current state, proposal scale) for each β_j, a truncated Normal(current state, proposal scale) between 0.51 and 0.99 for H, and fianlly a Normal(log(current state), proposal scale on a log scale) for σ^2, i.e., σ^2 is updated on a log scale. We sample the full posterior using Metropolis within Gibbs sampler. It also adopts adaptive Markov chain Monte Carlo (MCMC) that updates the proposal scales every 100 iterations; if the acceptance rate of the most recent 100 proposals (at the end of the ith 100 iterationsis) smaller than 0.3 then it multiplies $\exp(-\min(0.01, 1/\sqrt{i}))$ by the current proposal scale; if it is larger than 0.3 then it multiplies $\exp(\min(0.01, 1/\sqrt{i}))$ by the current proposal scale. The Markov chains equipped with this adaptive MCMC converge to the stationary distribution because the adjustment factors, $\exp(\pm\min(0.01, 1/\sqrt{i}))$, approach unity as i goes to infinity, satisfying the diminishing adaptation condition. The function `carfima` becomes numerically unstable if $p > 2$, and thus it may produce numerical errors. The output returns the AIC for which we evaluate the log likelihood at the posterior medians of the unknown model parameters.

If the MLE-based method produces numerical errors, we recommend running the Bayesian method that is numerically more stable (though computationally more expensive).

Value

A name list composed of:

mle If `method` is "mle". Maximum likelihood estimates of the model parameters, p α_j's, q β_j's, H, and σ.

se If `method` is "mle". Asymptotic uncertainties (standard errors) of the MLEs.

param If `method` is "bayes". An m by $(p+q+2)$ matrix where m is the number of posterior draws (`bayes.n.sample`) and the columns correspond to parameters, p α_j's, q β_j's, H, and σ.

accept If `method` is "bayes". A vector of length $p+q+2$ for the acceptance rates of the Metropolis steps.

AIC For both methods. Akaike Information Criterion, -2(log likelihood $-p-q-2$). The log likelihood is evaluated at the MLEs if `method` is "mle" and at the posterior medians of the unknown model parameters if `method` is "bayes".

fitted.values For both methods. A vector of length k for the values of $E(Y_{t_i}|Y_{<t_i})$, $i = 1, 2, \ldots, k$, where k is the number of observations and $Y_{<t_i}$ represents all data observed before t_i. Note that $E(Y_{t_1}|Y_{<t_1}) = 0$. MLEs of the model parameters are used to compute $E(Y_{t_i}|Y_{<t_i})$'s if `method` is "mle" and posterior medians of the model parameters are used if `method` is "bayes".

Details

The function `carfima` produces MLEs, their asymptotic uncertainties, and AIC if `method` is "mle". It produces the posterior samples of the model parameters, acceptance rates, and AIC if `method` is "bayes".

References

Tsai H, Chan KS (2000). "A NOTE ON THE COVARIANCE STRUCTURE OF A CONTINUOUS-TIME ARMA PROCESS." *Statistica Sinica*, **10**(3), 989–998.

Tsai H, Chan KS (2005). "Maximum likelihood estimation of linear continuous time long memory processes with discrete time data." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67**(5), 703–716. ISSN 1369-7412, 1467-9868, doi: 10.1111/j.14679868.2005.00522.x.

Examples

```
##### Irregularly spaced observation time generation.

length.time <- 100
time.temp <- rexp(length.time, rate = 2)
time <- rep(NA, length.time + 1)
time[1] <- 0
for (i in 2 : (length.time + 1)) {
  time[i] <- time[i - 1] + time.temp[i - 1]
}
time <- time[-1]

##### Data genration for CARFIMA(1, H, 0) based on the observation times.
```

```
parameter <- c(-0.4, 0.75, 0.2)
# AR parameter alpha = -0.4
# Hurst parameter = 0.75
# process uncertainty sigma = 0.2
y <- carfima.sim(parameter = parameter, time = time, ar.p = 1, ma.q = 0)

#### Estimation 1 : MLE

res1 <- carfima(Y = y, time = time, method = "mle", ar.p = 1, ma.q = 0)

#### Estimation 2 : Bayes

res2 <- carfima(Y = y, time = time, method = "bayes", ar.p = 1, ma.q = 0,
bayes.param.ini = parameter, bayes.param.scale = c(rep(0.2, length(parameter))),
bayes.n.warm = 100, bayes.n.sample = 1000)
```

`carfima.loglik` *Computing the log likelihood function of a CARFIMA(p, H, q) model*

Description

Computing the log likelihood function of a CARFIMA(p, H, q) model

Usage

```
carfima.loglik(Y, time, ar.p, ma.q, parameter, fitted = FALSE)
```

Arguments

`Y`	A vector for the k observed data.
`time`	A vector for the k observation times.
`ar.p`	A positive integer for the order of the AR model. `ar.p` must be greater than `ma.q`. If `ar.p` is greater than 2, numerical errors may occur for both methods.
`ma.q`	A non-negative integer for the order of the MA model. `ma.q` must be smaller than `ar.p`.
`parameter`	The values of the unknown parameters at which the log likelihood is evaluated. For example, users need to specify five values, α_1, α_2, β_1, H, and σ for `CARFIMA(2,H,1)`.
`fitted`	If `TRUE`, fitted values and AIC are returned as a list. `FALSE` flag returns a value of the log likelihood.

Value

Either a list of fitted values(`fitted`) and AIC(`AIC`), or a numeric value of the log likelihood.

Details

The function `carfima.loglik` computes the log likelihood of a `CARFIMA(p,H,q)` model via the innovation algorithm whose computational cost increases linearly as the size of the data increases. See the reference for details.

References

Tsai H, Chan KS (2000). "A NOTE ON THE COVARIANCE STRUCTURE OF A CONTINUOUS-TIME ARMA PROCESS." *Statistica Sinica*, **10**(3), 989–998.

Tsai H, Chan KS (2005). "Maximum likelihood estimation of linear continuous time long memory processes with discrete time data." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67**(5), 703–716. ISSN 1369-7412, 1467-9868, doi: 10.1111/j.14679868.2005.00522.x.

Examples

```
##### Irregularly spaced observation time generation.
length.time <- 100
time.temp <- rexp(length.time, rate = 2)
time <- rep(NA, length.time + 1)
time[1] <- 0
for (i in 2 : (length.time + 1)) {
  time[i] <- time[i - 1] + time.temp[i - 1]
  }
  time <- time[-1]

##### Data genration for CARFIMA(1, H, 0) based on the observation times.
parameter <- c(-0.4, 0.75, 0.2)
# AR parameter alpha = -0.4
# Hurst parameter = 0.75
# process uncertainty (standard deviation) sigma = 0.2
y <- carfima.sim(parameter = parameter, time = time, ar.p = 1, ma.q = 0)

##### Compute
output = carfima::carfima.loglik(Y=y,time=time,ar.p=1,ma.q=0,parameter=parameter,fitted=TRUE)
```

`carfima.sim` *Simulating a CARFIMA(p,H,q) time series*

Description

The function `carfima.sim` produces discrete time series data that follow a `CARFIMA(p,H,q)` model given the values for the model parameters and observation times.

Usage

```
carfima.sim(parameter, time, ar.p, ma.q)
```

Arguments

`parameter`	A vector of length $p + q + 2$ for the generative values of the model parameters; p values of α_j's, q values of β_j's, H and σ.
`time`	A vector for the k observation times, either regularly or irregularly spaced.
`ar.p`	A scalar for the order of the AR model.
`ma.q`	A scalar for the order of the MA model.

Value

The outcome of `carfima.sim` is a vector for k simulated data following a `CARFIMA(p,H,q)` model given the values for the model parameters and observation times.

Details

This function produces simulated discrete time series data following a `CARFIMA(p,H,q)` model given the values for the model parameters and observation times. It first derives a k-dimensional multivariate Gaussian distribution whose mean set to a vector of zeroes, where k is the number of observations. The covariance matrix is then filled with $Cov(Y_{t_i}, Y_{t_j})$ and its closed-form formula is specified in Theorem 1(b) and 1(c) of Tsai and Chan (2005).

References

Tsai H, Chan KS (2005). "Maximum likelihood estimation of linear continuous time long memory processes with discrete time data." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67**(5), 703–716. ISSN 1369-7412, 1467-9868, doi: 10.1111/j.14679868.2005.00522.x.

Examples

```
##### Irregularly spaced observation time generation.
##### For CRAN testing, time is set to be very short.

length.time <- 10
time.temp <- rexp(length.time, rate = 2)
time <- rep(NA, length.time + 1)
time[1] <- 0
for (i in 2 : (length.time + 1)) {
  time[i] <- time[i - 1] + time.temp[i - 1]
}
time <- time[-1]

##### Data genration for CARFIMA(1, H, 0) based on the observation times.

parameter <- c(-0.4, 0.75, 0.2)
# AR parameter alpha = -0.4
# Hurst parameter = 0.75
# process uncertainty sigma = 0.2
y <- carfima.sim(parameter = parameter, time = time, ar.p = 1, ma.q = 0)
```

package-carfima *Continuous-Time Fractionally Integrated ARMA Process for Irregularly Spaced Long-Memory Time Series Data*

Description

The R package **carfima** provides a toolbox to fit a continuous-time fractionally integrated ARMA process (CARFIMA) on univariate and irregularly spaced time series data via frequentist or Bayesian machinery. A general-order CARFIMA(p, H, q) model for $p > q$ is specified in Tsai and Chan (2005) and it involves $p + q + 2$ unknown model parameters, i.e., p AR parameters, q MA parameters, Hurst parameter H, and process uncertainty (standard deviation) σ; see also `carfima`. The package produces their maximum likelihood estimates and asymptotic uncertainties using a global optimizer called the differential evolution algorithm. It also produces their posterior distributions via Metropolis within a Gibbs sampler equipped with adaptive Markov chain Monte Carlo for posterior sampling. These fitting procedures, however, may produce numerical errors if $p > 2$. The toolbox also contains a function to simulate discrete time series data from CARFIMA(p, H, q) process given the model parameters and observation times.

Details

Package:	carfima
Type:	Package
Version:	2.0.0
License:	GPL-2
Main functions:	`carfima`, `carfima.loglik`, `carfima.sim`

References

Tsai H, Chan KS (2000). "A NOTE ON THE COVARIANCE STRUCTURE OF A CONTINUOUS-TIME ARMA PROCESS." *Statistica Sinica*, **10**(3), 989–998.

Tsai H, Chan KS (2005). "Maximum likelihood estimation of linear continuous time long memory processes with discrete time data." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67**(5), 703–716. ISSN 1369-7412, 1467-9868, doi: 10.1111/j.14679868.2005.00522.x.

Examples

```
##### Irregularly spaced observation time generation.

length.time <- 10
time.temp <- rexp(length.time, rate = 2)
time <- rep(NA, length.time + 1)
time[1] <- 0
```

```
for (i in 2 : (length.time + 1)) {
  time[i] <- time[i - 1] + time.temp[i - 1]
 }
 time <- time[-1]

 ##### Data genration for CARFIMA(1, H, 0) based on the observation times.

 parameter <- c(-0.4, 0.75, 0.2)
 # AR parameter alpha = -0.4
 # Hurst parameter = 0.75
 # process uncertainty (standard deviation) sigma = 0.2
 y <- carfima.sim(parameter = parameter, time = time, ar.p = 1, ma.q = 0)

 ##### Fitting the CARFIMA(1, H, 0) model on the simulated data for MLEs.

 res <- carfima(Y = y, time = time, method = "mle", ar.p = 1, ma.q = 0)

 # It takes a long time due to the differential evolution algorithm (global optimizer).
 # res$mle; res$se; res$AIC; res$fitted.values

 ##### Fitting the CARFIMA(1, H, 0) model on the simulated data for Bayesian inference.
 res <- carfima(Y = y, time = time, method = "bayes",
                ar.p = 1, ma.q = 0,
                bayes.param.ini = parameter,
                bayes.param.scale = c(rep(0.2, length(parameter))),
                bayes.n.warm = 100, bayes.n.sample = 1000)

 # It takes a long time because the likelihood evaluation is computationally heavy.
 # The last number of bayes.param.scale is to update sigma2 (not sigma) on a log scale.
 # hist(res$param[, 1]); res$accept; res$AIC; res$fitted.values

 ##### Computing the log likelihood of the CARFIMA(1, H, 0) model given the parameters.

 loglik <- carfima.loglik(Y = y, time = time, ar.p = 1, ma.q = 0,
                          parameter = parameter, fitted = FALSE)
```

Index