Package: carfima (via r-universe)

August 22, 2024

Version 2.0.2

Date 2020-03-20

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

Author Hyungsuk Tak, Henghsiu Tsai, and Kisung You

Maintainer Hyungsuk Tak <hyungsuk.tak@gmail.com>

Depends R (>= 2.2.0)

Imports mvtnorm(>= 1.0-11), DEoptim (>= 2.2-5), pracma (>= 2.2.9), truncnorm (>= 1.0-8), invgamma (>= 1.1)

Description We provide a toolbox to fit a continuous-time fractionally integrated ARMA process (CARFIMA) on univariate and irregularly spaced time series data via both frequentist and 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. Also, the model can account for heteroscedastic measurement errors, if the information about measurement error standard deviations is known. The package produces their maximum likelihood estimates and asymptotic uncertainties using a global optimizer called the differential evolution algorithm. It also produces posterior samples of the model parameters via Metropolis-Hastings within a Gibbs sampler equipped with adaptive Markov chain Monte Carlo. 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

NeedsCompilation no

Date/Publication 2020-03-21 17:50:02 UTC

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

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

RemoteRef HEAD

RemoteSha 84b56e8642c5350fcdf9fb013592ffdcd78240aa

Contents

`carfima-package` *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 both frequentist and Bayesian machinery. A general-order CARFIMA(p, H, q) model for $p > q$ is specified in Tsai and Chan (2005). 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`. Also, the model can account for heteroscedastic measurement errors, if the information about measurement error standard deviations is known. The package produces their maximum likelihood estimates and asymptotic uncertainties using a global optimizer called the differential evolution algorithm. It also produces posterior samples of the model parameters via Metropolis-Hastings within a Gibbs sampler equipped with adaptive Markov chain Monte Carlo. 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:	2.0.2
Date:	2020-03-20
License:	GPL-2
Main functions:	`carfima`, `carfima.loglik`, `carfima.sim`

Author(s)

Hyungsuk Tak, Henghsiu Tsai, and Kisung You

Maintainer: Hyungsuk Tak <hyungsuk.tak@gmail.com>

References

H. Tsai and K.S. Chan (2005) "Maximum Likelihood Estimation of Linear Continuous Time Long Memory Processes with Discrete Time Data," Journal of the Royal Statistical Society (Series B), 67 (5), 703-716. DOI: 10.1111/j.1467-9868.2005.00522.x

H. Tsai and K.S. Chan (2000) "A Note on the Covariance Structure of a Continuous-time ARMA Process," Statistica Sinica, 10, 989-998.
Link: http://www3.stat.sinica.edu.tw/statistica/j10n3/j10n317/j10n317.htm

Examples

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

length.time <- 30
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, 0.8, 0.2)
# AR parameter alpha = -0.4
# Hurst parameter = 0.8
# Process uncertainty (standard deviation) sigma = 0.2

me.sd <- rep(0.05, length.time)
# Known measurement error standard deviations 0.05 for all observations
# If not known, remove the argument "measure.error = me.sd" in the following codes,
# so that the default values (zero) are automatically assigned.

y <- carfima.sim(parameter = parameter, time = time,
                 measure.error = me.sd, 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, measure.error = me.sd,
               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, measure.error = me.sd,
               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,
                         measure.error = me.sd,
                         parameter = parameter, fitted = FALSE)
```

`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. Also, the model can account for heteroscedastic measurement errors, if the information about measurement error standard deviations is known.

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, (0, 100) 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 **pracma**). 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 Bayesian approach uses independent prior distributions for the unknown model parameters; a Uniform(-0.9999, -0.0001) prior for each α_j, a Uniform(0, 100) prior for each β_j, a Uniform(0.5001, 0.9999) 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 -0.9999 and -0.0001 for each α_j, a truncated Normal(current state, proposal scale) between 0

and 100 for each β_j, a truncated Normal(current state, proposal scale) between 0.5001 and 0.9999 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-Hastings 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 iterations) smaller than 0.3, then it multiplies $\exp(-\min(0.1, 1/\sqrt{i}))$ by the current proposal scale; if it is larger than 0.3, then it multiplies $\exp(\min(0.1, 1/\sqrt{i}))$ by the current proposal scale. The resulting Markov chain with this adaptive scheme converge to the stationary distribution because the adjustment factor, $\exp(\pm\min(0.1, 1/\sqrt{i}))$, approaches 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.

Usage

```
carfima(Y, time, measure.error, ar.p, ma.q, method = "mle",
              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.
`measure.error`	(Optional) A vector for the k measurement error standard deviations, if such information is available (especially for astronomical applications). A vector of zeros is automatically assigned, if nothing is specified.
`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-Hastings 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-Hastings steps. Each number determines how far a Metropolis-Hastings 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 `carfiam` 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".

Value

The outcome of `carfima` is 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-Hastings 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".

Author(s)

Hyungsuk Tak, Henghsiu Tsai, Kisung You

References

H. Tsai and K.S. Chan (2005) "Maximum Likelihood Estimation of Linear Continuous Time Long Memory Processes with Discrete Time Data," Journal of the Royal Statistical Society (Series B), 67 (5), 703-716. DOI: 10.1111/j.1467-9868.2005.00522.x

H. Tsai and K.S. Chan (2000) "A Note on the Covariance Structure of a Continuous-time ARMA Process," Statistica Sinica, 10, 989-998.
Link: http://www3.stat.sinica.edu.tw/statistica/j10n3/j10n317/j10n317.htm

Examples

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

length.time <- 30
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, 0.8, 0.2)
# AR parameter alpha = -0.4
# Hurst parameter = 0.8
# Process uncertainty (standard deviation) sigma = 0.2

me.sd <- rep(0.05, length.time)
# Known measurement error standard deviations 0.05 for all observations
# If not known, remove the argument "measure.error = me.sd" in the following codes,
# so that the default values (zero) are automatically assigned.

y <- carfima.sim(parameter = parameter, time = time,
                 measure.error = me.sd, 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, measure.error = me.sd,
               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, measure.error = me.sd,
               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,
                         measure.error = me.sd,
                         parameter = parameter, fitted = FALSE)
```

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

Description

This function evaluates the log likelihood function of a CARFIMA(p, H, q) model as specified in Tsai and Chan (2005).

Usage

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

Arguments

`Y`	A vector for the k observed data.
`time`	A vector for the k observation times.
`measure.error`	(Optional) A vector for the k measurement error standard deviations, if such information is available (especially for astronomical applications). A vector of zeros is automatically assigned, if nothing is specified.
`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 of α_1, α_2, β_1, H, and σ for CARFIMA(2, H, 1).
`fitted`	If "TRUE", fitted values and AIC are returned. If "FALSE", a value of the log likelihood is returned. Default is "FALSE".

Details

The function `carfiam.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 Tsai and Chan (2005) for details.

Value

The outcome of `carfima` is the value of the log likelihood if "fitted = FALSE" and both AIC and fitted values if "fitted = TRUE".

Author(s)

Hyungsuk Tak, Henghsiu Tsai, Kisung You

References

H. Tsai and K.S. Chan (2005) "Maximum Likelihood Estimation of Linear Continuous Time Long Memory Processes with Discrete Time Data," Journal of the Royal Statistical Society (Series B), 67 (5), 703-716. DOI: 10.1111/j.1467-9868.2005.00522.x

H. Tsai and K.S. Chan (2000) "A Note on the Covariance Structure of a Continuous-time ARMA Process," Statistica Sinica, 10, 989-998.
Link: http://www3.stat.sinica.edu.tw/statistica/j10n3/j10n317/j10n317.htm

Examples

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


length.time <- 30
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, 0.8, 0.2)
# AR parameter alpha = -0.4
# Hurst parameter = 0.8
# Process uncertainty (standard deviation) sigma = 0.2

me.sd <- rep(0.05, length.time)
# Known measurement error standard deviations 0.05 for all observations
# If not known, remove the argument "measure.error = me.sd" in the following codes,
# so that the default values (zero) are automatically assigned.

y <- carfima.sim(parameter = parameter, time = time,
                 measure.error = me.sd, ar.p = 1, ma.q = 0)

##### 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,
                         measure.error = me.sd,
                         parameter = parameter, fitted = FALSE)
```

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

Description

The funstion `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, measure.error, 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.
measure.error	(Optional) A vector for the k measurement error standard deviations, if such information is available (especially for astronomical applications). A vector of zeros is automatically assigned, if nothing is specified.
ar.p	A scalar for the order of the AR model.
ma.q	A scalar for the order of the MA model.

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 zeros, where k is the number of observations. The covariance matrix is 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).

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.

Author(s)

Hyungsuk Tak, Henghsiu Tsai, Kisung You

References

H. Tsai and K.S. Chan (2005) "Maximum Likelihood Estimation of Linear Continuous Time Long Memory Processes with Discrete Time Data," Journal of the Royal Statistical Society (Series B), 67 (5), 703-716. DOI: 10.1111/j.1467-9868.2005.00522.x

Examples

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

length.time <- 30
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, 0.8, 0.2)
# AR parameter alpha = -0.4
# Hurst parameter = 0.8
# Process uncertainty (standard deviation) sigma = 0.2
```

```
me.sd <- rep(0.05, length.time)
# Known measurement error standard deviations 0.05 for all observations
# If not known, remove the argument "measure.error = me.sd" in the following codes,
# so that the default values (zero) are automatically assigned.

y <- carfima.sim(parameter = parameter, time = time,
                 measure.error = me.sd, ar.p = 1, ma.q = 0)
```

Index