Package 'garma'

July 26, 2020

Type Package

Title GARMA provides support for estimating, and fitting Gegenbauer Seasonal/Cyclical time series models

Version 0.9.0

Date 2020-07-26

Maintainer Richard Hunt<maint@huntemail.id.au>

Description This package provides methods for estimating long memory-seasonal/cyclical Gegenbauer univariate time series proceses.
Refer to the vignette for details of fitting these processes.

License GPL-3

Encoding UTF-8

LazyData true

Imports assertthat,
zoo,
forecast,
lubridate,
FKF,
signal,
pracma,
nloptr,
Rsolnp,
ggplot2,
Rdpack (>= 0.7)

Suggests longmemo,
tidyverse,
BB,
GA,
pso,
dfoptim,
testthat,
knitr,
rmarkdown

RoxygenNote 7.0.1

VignetteBuilder knitr

R topics documented:

`extract_arma` *For a k=1 Gegenbauer process, transform to remove Gegenbauer long memory component to get a short memory (ARMA) process.*

Description

For a k=1 Gegenbauer process, transform to remove Gegenbauer long memory component to get a short memory (ARMA) process.

Usage

```
extract_arma(x, ggbr_factors)
```

Arguments

`x`	(num) This should be a numeric vector representing the process to estimate.
`ggbr_factors`	(list) Each element of the list represents a Gegenbauer factor and includes f, u and fd elements.

Value

An object of same class as x.

Examples

```
data(AirPassengers)
ap <- as.numeric(diff(AirPassengers,12))
# find semiparametric estimates of the Gegenbauer parameters.
sp <- ggbr_semipara(ap)
# extract the underlying short-memory ARMA process
ap_arma <- extract_arma(ap,sp$ggbr_factors)
summary(arima(ap_arma,order=c(1,0,0)))
```

`forecast.garma_model` *The forecast function predicts future values of a "garma_model" object, and is exactly the same as the "predict" function with slightly different parameter values.*

Description

The forecast function predicts future values of a "garma_model" object, and is exactly the same as the "predict" function with slightly different parameter values.

Usage

```
## S3 method for class 'garma_model'
forecast(mdl, h = 1)
```

Arguments

`mdl`	(garma_model) The garma_model from which to forecast the values.
`h`	(int) The number of time periods to predict ahead. Default: 1

Value

- a "ts" object containing the requested forecasts.

Examples

```
library(forecast)

data(AirPassengers)
ap  <- as.numeric(diff(AirPassengers,12))
mdl <- garma(ap,order=c(9,1,0),k=0,method='CSS',include.mean=FALSE)
forecast(mdl, h=12)
```

`garma` *garma: A package for estimating and foreasting Gegenbauer time series models.*

Description

The GARMA package provides the main function "garma" as well as print, summary, predict, forecast and plot/ggplot options.

The garma function is the main function for the garma package. Depending on the parameters it will calculate the parameter estimates for the GARMA process, and if available the standard errors (se's) for those parameters.

Usage

```
garma(
  x,
  order = list(0, 0, 0),
  k = 1,
  include.mean = (order[2] == 0),
  method = "Whittle",
  allow_neg_d = FALSE,
  maxeval = 10000,
  opt_method = NULL,
  m_trunc = 50,
  min_freq = 0,
  max_freq = 0.5
)
```

Arguments

x	(num) This should be a numeric vector representing the process to estimate. A minimum length of 96 is required.
order	(list) This should be a list (similar to the stats::arima order parameter) which will give the order of the process to fit. The format should be list(p,d,q) where p, d, and q are all positive integers. p represents the degree of the autoregressive process to fit, q represents the order of the moving average process to fit and d is the (integer) differencing to apply prior to any fitting.
k	(int) This is the number of (multiplicative) Gegenbauer terms to fit.
include.mean	(bool) A boolean value indicating whether a mean should be fit. Note if you have any differencing, then it generally does not make sense to fit a mean term. Because of this, the default here is to fit a mean time when d (in the "order" parmaeter) is zero and otherwise not.
method	(character) This defines the estimation method for the routine. The valid values are 'CSS', 'Whittle', 'QML' and 'WLL'. The default (Whittle) will generally return very accurate estimates quite quickly, provided the asumption of a Gaussian distribution is even approximately correct, and is probably the method of choice for most users. For the theory behind this, refer Giraitis et. al. (2001) 'CSS' is a conditional 'sum-of-squares' technique and can be quite slow. Reference: Chung (1996). 'QML' is a Quasi-Maximum-Likelihood technique, and can also be quite slow. Reference Dissanayake (2016). (k>1 is not supported for QML) 'WLL' is a new technique which appears to work well even if the ϵ_t are highly skewed and/or have heavy tails (skewed and/or lepto-kurtic). However the asymptotic theory for the WLL method is not complete and so standard errors are not available for most parameters.
allow_neg_d	(bool) A boolean value indicating if a negative value is allowed for the fractional differencing component of the Gegenbauer term is allowed. This can be set to FALSE (the default) to force the routine to find a positive value.
maxeval	(int) the maximum function eveluations to be allowed during each optimisation.

`opt_method` (character) This names the optimisation method used to find the parameter estimates. This may be a list of methods, in which case the methods are applied in turn, each using the results of the previous one as the starting point for the next. The default is to use c('directL', 'solnp') when k<2 and 'solnp' when k>=2. The directL algorithm is used to perform a global search for the minima, and solnp to refine the values. For some data or some models, however, other methods may work well. Supported algorithms include:

- cobyla algorithm in package nloptr
- directL algorithm in package nloptr
- BBoptim from package BB
- psoptim from package pso
- hjkb from dfoptim package
- nmkb from dfoptim package
- solnp from Rsolnp package
- best - this option evaluates all the above options in turn and picks the one which finds the lowest value of the objective. This can be quite time consuming to run, particularly for the 'CSS' method.

Note further that if you specify a k>1, then inequality constraints are required, and this will further limit the list of supported routines.

`m_trunc` Used for the QML estimation method. This defines the AR-truncation point when evaluating the likelihood function. Refer to Dissanayake et. al. (2016) for details.

`min_freq` (num) When searching for Gegenbauer peaks, this is the minimum frequency used. Default 0. Note that when there is an AR(1) component, the peaks corresponding to the AR(1) can be higher than the Gegenbauer peaks. Setting this parameter to 0.05 or above can help.

`max_freq` (num) default 0.5. When searching for Gegenbauer peaks, this is the maximum frequency used.

Details

The GARMA model is specified as

$$\phi(B)\prod_{i=1}^{k}(1-2u_iB+B^2)^{d_i}(X_t-\mu)=\theta(B)\epsilon_t$$

where

- $\phi(B)$ represents the short-memory Autoregressive component of order p,
- $\theta(B)$ represents the short-memory Moving Average component of order q,
- $(1-2u_iB+B^2)^{d_i}$ represents the long-memory Gegenbauer component (there may in general be k of these),
- X_t represents the observed process,
- ϵ_t represents the random component of the model - these are assumed to be uncorrelated but identically distributed variates. Generally the routines in this package will work best if these have an approximate Gaussian distribution.

- B represents the Backshift operator, defined by $BX_t = X_{t-1}$.

when k=0, then this is just a short memory model as fit by the stats "arima" function.

Value

An S3 object of class "garma_model".

Author(s)

Richard Hunt

References

C Chung. A generalized fractionally integrated autoregressive moving-average process. *Journal of Time Series Analysis*, **17**(2):111–140, 1996.

G Dissanayake, S Peiris, and T Proietti. State space modelling of Gegenbauer processes with long memory. *Computational Statistics and Data Analysis*, **100**:115–130, 2016.

L Giraitis, J Hidalgo, and P Robinson. Gaussian estimation of parametric spectral density with unknown pole. *The Annals of Statistics*, **29**(4):987–1023, 2001.

Examples

```
data(AirPassengers)
ap  <- as.numeric(diff(AirPassengers,12))
print(garma(ap,order=c(9,1,0),k=0,method='CSS',include.mean=FALSE))
# Compare with the built-in arima function
print(arima(ap,order=c(9,1,0),include.mean=FALSE))
```

---

garma_ggtsdisplay — *For a k=1 Gegenbauer process, use semi-parametric methods to obtain short memory version of the process, then run a ggtsdisplay().*

Description

For a k=1 Gegenbauer process, use semi-parametric methods to obtain short memory version of the process, then run a ggtsdisplay().

Usage

```
garma_ggtsdisplay(x, k = 1, ...)
```

Arguments

x	(num) This should be a numeric vector representing the process to estimate.
k	(int) The number of Gegenbauer factors
...	additional parameters to pass to ggtsdisplay

Value

A ggplot object.

Examples

```
data(AirPassengers)
ap <- as.numeric(diff(AirPassengers,12))
garma_ggtsdisplay(ap)
```

`ggbr_semipara` *For a k=1 Gegenbauer process, use semi-parametric methods to estimate the Gegenbauer frequency and fractional differencing.*

Description

For a k=1 Gegenbauer process, use semi-parametric methods to estimate the Gegenbauer frequency and fractional differencing.

Usage

```
ggbr_semipara(
  x,
  k = 1,
  alpha = 0.8,
  method = "gsp",
  min_freq = 0,
  max_freq = 0.5
)
```

Arguments

`x`	(num) This should be a numeric vector representing the process to estimate.
`k`	(int) The number of Gegenabuer frequencies
`alpha`	(num)
`method`	(char) One of "gsp" or "lpr" - lpr is the log-periodogram-regression technique, "gsp" is the Gaussian semi-parametric technique. "gsp" is the default. Refer Arteche (1998).
`min_freq`	(num) The minimum frequency to search through for peaks - default 0.0.
`max_freq`	(num) The maximum frequency to search through for peaks - default 0.5.

Value

An object of class "garma_semipara".

Examples

```
data(AirPassengers)
ap <- as.numeric(diff(AirPassengers,12))
sp <- ggbr_semipara(ap)
print(sp)
```

`ggplot.garma_model` *The ggplot function generates a ggplot of actuals and predicted values for a "garma_model" object.*

Description

The ggplot function generates a ggplot of actuals and predicted values for a "garma_model" object.

Usage

```
## S3 method for class 'garma_model'
ggplot(mdl, h = 24, ...)
```

Arguments

`mdl`	(garma_model) The garma_model from which to ggplot the values.
`h`	(int) The number of time periods to predict ahead. Default: 24
`...`	other parameters passed to ggplot.

Value

A ggplot2 "ggplot" object. Note that the standard ggplot2 "+" notation can be used to enhance the default output.

Examples

```
library(ggplot2)

data(AirPassengers)
ap  <- as.numeric(diff(AirPassengers,12))
mdl <- garma(ap,order=c(9,1,0),k=0,method='CSS',include.mean=FALSE)
ggplot(mdl)
```

gg_raw_pgram *Display the raw periodogram for a time series, not on a log scale. The standard "R" functions display periodograms on a log scale which can make it more difficult to locate high peaks in the spectrum at differning frequencies. This routine will display the peaks on a raw scale.*

Description

Display the raw periodogram for a time series, not on a log scale. The standard "R" functions display periodograms on a log scale which can make it more difficult to locate high peaks in the spectrum at differning frequencies. This routine will display the peaks on a raw scale.

Usage

```
gg_raw_pgram(x, k = 1)
```

Arguments

x	(num) This should be a numeric vector representing the process to estimate.
k	(int) The number of Gegenbauer factors

Value

A ggplot object representing the raw periodogram

Examples

```
data(AirPassengers)
ap <- as.numeric(diff(AirPassengers,12))
sp <- ggbr_semipara(ap)
print(sp)
```

plot.garma_model *The plot function generates a plot of actuals and predicted values for a "garma_model" object.*

Description

The plot function generates a plot of actuals and predicted values for a "garma_model" object.

Usage

```
## S3 method for class 'garma_model'
plot(x, ...)
```

Arguments

x	(garma_model) The garma_model from which to plot the values.
...	other arguments to be passed to the "plot" function, including h (int) - the number of periods ahgead to forecast.

Value

An R "plot" object.

Examples

```
data(AirPassengers)
ap  <- as.numeric(diff(AirPassengers,12))
mdl <- garma(ap,order=c(9,1,0),k=0,method='CSS',include.mean=FALSE)
plot(mdl)
```

---

predict.garma_model *The predict function predicts future values of a "garma_model" object.*

Description

The predict function predicts future values of a "garma_model" object.

Usage

```
## S3 method for class 'garma_model'
predict(object, n.ahead = 1, ...)
```

Arguments

object	(garma_model) The garma_model from which to predict the values.
n.ahead	(int) The number of time periods to predict ahead. Default: 1
...	Other parameters. Ignored.

Value

A "ts" object containing the requested forecasts.

Examples

```
data(AirPassengers)
ap  <- as.numeric(diff(AirPassengers,12))
mdl <- garma(ap,order=c(9,1,0),k=0,method='CSS',include.mean=FALSE)
predict(mdl, n.ahead=12)
```

print.garma_model *The print function prints a summary of a "garma_model" object.*

Description

The print function prints a summary of a "garma_model" object.

Usage

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

Arguments

x	(garma_model) The garma_model from which to print the values.
...	Other arguments. Ignored.

Examples

```
data(AirPassengers)
ap  <- as.numeric(diff(AirPassengers,12))
mdl <- garma(ap,order=c(9,1,0),k=0,method='CSS',include.mean=FALSE)
print(mdl)
```

print.garma_semipara *Print a semiparameteric Gegenbauer estimation object.*

Description

Print a semiparameteric Gegenbauer estimation object.

Usage

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

Arguments

x	An object of class garma_semipara.
...	further parameters for print function

print.ggbr_factors *Print a 'ggbr_factors' object.*

Description

Print a 'ggbr_factors' object.

Usage

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

Arguments

x	An object of class ggbr_factors
...	further parameters for print function

summary.garma_model *The summary function provides a summary of a "garma_model" object.*

Description

The summary function provides a summary of a "garma_model" object.

Usage

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

Arguments

object	(garma_model) The garma_model from which to print the values.
...	Other arguments. Ignored.

Examples

```
data(AirPassengers)
ap  <- as.numeric(diff(AirPassengers,12))
mdl <- garma(ap,order=c(9,1,0),k=0,method='CSS',include.mean=FALSE)
summary(mdl)
```

Index