Package: garma (via r-universe)

September 13, 2024

Type Package

Title Fitting and Forecasting Gegenbauer ARMA Time Series Models

Version 0.9.23

Date 2024-09-13

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

Description Methods for estimating univariate long memory-seasonal/cyclical Gegenbauer time series processes. See for example (2022) <doi:10.1007/s00362-022-01290-3>. Refer to the vignette for details of fitting these processes.

License GPL-3

URL https://github.com/rlph50/garma

Encoding UTF-8

LazyData false

Depends forecast, ggplot2

Imports Rsolnp, nloptr, pracma, signal, zoo, lubridate, rlang, crayon, utils

Suggests longmemo, yardstick, testthat (>= 3.0.0), knitr, rmarkdown

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://rlph50.r-universe.dev

RemoteUrl https://github.com/rlph50/garma

RemoteRef HEAD

RemoteSha ef1f3c2fe7ffce58ed65f9725136598930e17008

Contents

`AIC.garma_model` *AIC for model*

Description

Approximate AIC for model.

Usage

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

Arguments

`object`	The garma_model object
`...`	Other parameters. Ignored.

Value

(double) Approximate AIC - uses approximation of whichever methoid is used to find model params.

autoplot.garma_model *ggplot of the Forecasts of the model.*

Description

The ggplot function generates a ggplot of actuals and predicted values for a "garma_model" object. This adds in sensible titles etc as best it can determine.

Usage

```
## S3 method for class 'garma_model'
autoplot(object, h = 24, include_fitted = FALSE, ...)
```

Arguments

object	(garma_model) The garma_model from which to ggplot the values.
h	(int) The number of time periods to predict ahead. Default: 24
include_fitted	(bool) whether to include the 1-step ahead 'fitted' values in the plot. Default: FALSE
...	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)
autoplot(mdl)
```

coef.garma_model *Model Coefficients*

Description

Model Coefficients/parameters.

Usage

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

Arguments

object	The garma_model object
...	Other parameters. Ignored.

Value

(double) array of parameter value estimates from the fitted model.

---

extract_arma *Extract underlying ARMA process.*

Description

For a 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 Gegenbauer process.
ggbr_factors	(class) Each element of the list represents a Gegenbauer factor and includes f, u and fd elements.

Value

An object of same class as x. Any time series attributes of x are copied to the returned object.

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)
summary(arima(ap_arma, order = c(1, 0, 0)))
```

fitted.garma_model *Extract fitted values*

Description

Fitted values are 1-step ahead predictions.

Usage

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

Arguments

object	The garma_model object
...	Other parameters. Ignored.

Value

(double) array of 1-step ahead fitted values for the model.

forecast.garma_model *Forecast future 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(object, h = 1, newdata = NULL, ...)
```

Arguments

object	(garma_model) The garma_model from which to forecast the values.
h	(int) The number of time periods to predict ahead. Default: 1
newdata	(real vector or matrix) If the original model was fitted with the 'xreg=' option then this will provide the xreg values for predictions. If this is a vector then its length should be 'h'; if it is a matrix then it should have 'h' rows. It should have columns with the same names as the original xreg matrix.
...	Other parameters passed to the forecast function. For "garma_model" objects, these are ignored.

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 = c(0L, 0L, 0L),
  periods = NULL,
  k = 1,
  include.mean = (order[2] == 0L),
  include.drift = FALSE,
  xreg = NULL,
  method = "Whittle",
  d_lim = c(0, 0.5),
  opt_method = c("cobyla", "solnp"),
  control = NULL
)
```

Arguments

x	(num) This should be a numeric vector representing the process to estimate. A minimum length of 96 is required.

`order` (numeric vector) This should be a vector (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. WARNING: Currently only d==0 or d==1 are allowed.

`periods` (num) This parameter can be used to specify a fixed period or set of periods for the Gegenbauer periodicity. For instance if you have monthly data, then it might be sensible (after an examination of the periodogram) to set periods = 12. The default value is NULL. Either 'periods' or 'k' parameters must be specified but not both - 'periods' implies fixed period(s) are to be used and 'k' implies that the periods should be estimated.

`k` (int) This parameter indicates that the algorithm should estimate the 'k' frequencies as a part of the model. An alternative is the 'periods' parameter which can be used to specify exactly which periods should be used by the model.

This parameter can also be interpreted as specifying the number of (multiplicative) Gegenbauer terms to fit in the model.

`include.mean` (bool) A boolean value indicating whether a mean should be fit. Note that no mean term is fit if the series is integer differenced.

`include.drift` (bool) A boolean value indicating whether a 'drift' term should be fit to the predictions. The default is to fit a drift term to the predictions if the process is integer-differenced.

`xreg` (numeric matrix) A numerical vector or matrix of external regressors, which must have the same number of rows as x. It should not have any NA values. It should not be a data frame. The default value is NULL.

Note that the algorithm used here is that if any 'xreg' is supplied, then a linear regression model is fit first, and the GARMA model is then based on the residuals from that regression model.

`method` (character) This defines the estimation method for the routine. The valid values are 'CSS', 'Whittle', and 'WLL'. The default ('Whittle') method will generally return very accurate estimates quite quickly, provided the assumption 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).

The 'CSS' method is a conditional 'sum-of-squares' technique and can be quite slow. Reference: Robinson (2006), Chung (1996). Note that the paper of Chung (1996) was partially critisised by Giraitis et. al. (2001), however still contains useful results.

'WLL' is a new technique, originally developed by the author of this package and 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. Refer Hunt et. al. (2021).

`d_lim` (list) the limits for the d parameter. The default is 'c(0,0.5)', which restricts the model to be stationary. However sometimes it is desirable to understand what the unrestricted value might be.

`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('solnp', 'cobyla'). 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
- 'solnp' from Rsolnp package
- 'gosolnp' from Rsolnp package.

Note that the algorithms are selected to be those which do not require derivatives, even numerically calculated derivatives. The function being optimised by 'garma()' has a point of discontinuity at the minimum value - the point we are trying to find. This means that standard algorithms like BFGS et al. perform very poorly here.

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

`control` (list) list of optimisation routine specific values.

Details

The GARMA model is specified as

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

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),
- id represents the degree of integer differencing, where as d_i represents the degree of fractional differencing. Note that id is a value supplied by the user (the second number on the 'order=' parameter - similarly to the way that the base R 'arima' function works) whereas d_i is estimated by this function.
- 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. DOI: https://doi.org/10.1111/j.1467-9892.1996.tb00268.x

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. DOI: https://doi.org/10.1214/AOS/1013699989

R Hunt, S Peiris, N Webe. A General Frequency Domain Estimation Method for Gegenbauer Processes. Journal of Time Series Econometrics, 13(2):119-144, 2021. DOI: https://doi.org/10.1515/jtse-2019-0031

R Hunt, S Peiris, N Weber. Estimation methods for stationary Gegenbauer processes. Statistical Papers 63:1707-1741, 2022. DOI: https://doi.org/10.1007/s00362-022-01290-3

P. Robinson. Conditional-sum-of-squares estimation of models for stationary time series with long memory. IMS Lecture Notes Monograph Series, Time Series and Related Topics, 52:130-137, 2006. DOI: https://doi.org/10.1214/074921706000000996.

See Also

Useful links:

- https://github.com/rlph50/garma

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 *ggtsdisplay of underlying ARMA process.*

Description

For a 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

Details

The purpose of this function is to ease the process of identifying the underlying short memory process.

Value

A ggplot object.

Examples

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

---

ggbr_semipara *Extract semiparametric estimates of the Gegenbauer factors.*

Description

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

Usage

```
ggbr_semipara(x, periods = NULL, k = 1, alpha = 0.8, method = "gsp")
```

Arguments

x	(num) This should be a numeric vector representing the process to estimate.
periods	(num) This parameter can be used to specify a fixed period or set of periods for the Gegenbauer periodicity. For instance if you have monthly data, then it might be sensible (after an examination of the periodogram) to set 'periods = 12'. The default value is NULL. Either 'periods' or 'k' parameters must be specified but not both - 'periods' implies fixed period(s) are to be used and 'k' implies that the periods should be estimated.
k	(int) This parameter indicates that the algorithm should estimate the 'k' frequencies semi-parametrically, before estimating the degree of fractional differencing at each period. An alternative is the 'periods' parameter which can be used to specify exactly which periods should be used by the model.

alpha	(num) Default = 0.8 - This is the bandwidth for the semiparametric estimate, and should be between 0 and 1. Robinson (1994) indicated optimality for a (scaled) version of ‘alpha‘ = 0.8, at least for the "lpr" ‘method‘.
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 & Robinson (1998).

Value

An object of class "garma_semipara".

References

J Arteche and P Robinson. Semiparametric inference in seasonal and cyclical long memory processes. Journal of Time Series Analysis, 21(1):1–25, 2000. DOI: https://doi.org/10.1111/1467-9892.00170

P Robinson. Rates of convergence and optimal spectral bandwidth for long range dependence. Probability Theory and Related Fields, 99:443–473, 1994. DOI: https://doi.org/10.1007/BF01199901.

Examples

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

---

gg_raw_pgram *Display raw periodogram*

Description

Display the raw periodogram for a time series, and not on a log 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

Details

The standard "R" functions display periodograms on a log scale which can make it more difficult to locate high peaks in the spectrum at differing frequencies. This routine will display the peaks on a raw scale.

Value

A ggplot object representing the raw periodogram

Examples

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

`gof` *Goodness-of-Fit test for a garma_model.*

Description

Provides a goodness-of-fit test for a GARMA Model, using Bartletts Tp test. This has been justified for long memory and for GARMA models by Delgado, Hidalgo and Velasco (2005).

Usage

```
gof(object)
```

Arguments

`object`	(garma_model) The garma_model to test.

Value

Invisibly returns the array of p-values from the test.

References

M Delgado, J Hidalgo, and C Velasco. Distribution free goodness-of-fit tests for linear processes. The Annals of Statistics, 33(6):2568–2609, 2005. DOI: https://doi.org/10.1214/009053605000000606.

logLik.garma_model *Log Likelihood*

Description

The approximate likelihood for the model.

Usage

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

Arguments

object	The garma_model object
...	Other parameters. Ignored.

Value

Object of class "logLik" with values for the (approx) log-likelihood for the model

plot.garma_model *Plot Forecasts from model.*

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 ahead 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` *Predict future values.*

Description

Predict ahead using algorithm of Godet (2009).

Usage

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

Arguments

`object`	(garma_model) The garma_model from which to predict the values. This should have been generated by the [garma()] function.
`n.ahead`	(int) The number of time periods to predict ahead. Default: 1
`newdata`	(real vector or matrix) If the original model was fitted with the 'xreg=' option then this will provide the xreg values for predictions. If this is a vector then its length should be 'n.ahead'; if it is a matrix then it should have 'n.ahead' rows. It should have columns with the same names as the original xreg matrix.
`...`	Other parameters. Ignored.

Value

A "ts" object containing the requested forecasts.

References

Godet, F. Linear prediction of long-range dependent time series, ESAIM: PS (2009) 13 115-134. DOI: https://doi.org/10.1051/ps:2008015

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 *print a garma_model object.*

Description

The print function prints a summary of a "garma_model" object, printed to the output.

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.

Value

(null)

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.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

Value

null

residuals.garma_model *Residuals*

Description

Response Residuals from the model.

Usage

```
## S3 method for class 'garma_model'
residuals(object, type = "response", h = 1, ...)
```

Arguments

object	The garma_model object
type	(chr) The type of residuals. Must be 'response'.
h	(int) The number of periods ahead for the residuals. Must be 1.
...	Other parameters. Ignored.

Value

(double) array of resideuals from the model.

summary.garma_model *summarise a garma_model object.*

Description

The summary function provides a summary of a "garma_model" object, printed to the output.

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.

Value

(null)

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)
```

`tsdiag.garma_model` *Diagnostic fit of a garma_model.*

Description

Produces diagnostic plots of the model fit. This function is copied from stats::tsdiag but modifies the fit_df for the Ljung-Box test for use with garma models.

Usage

```
## S3 method for class 'garma_model'
tsdiag(object, gof.lag = 10, ...)
```

Arguments

`object`	(garma_model) The garma_model to produce the diagnostic plots for.
`gof.lag`	(int) The number of lags to examine for the Ljung-Box white noise test.
`...`	further arguments to be passed to particular methods.

Value

None. Diagnostics are generated.

See Also

The stats package tsdiag function: https://stat.ethz.ch/R-manual/R-patched/library/stats/html/tsdiag.html.

Examples

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

`vcov.garma_model` *Covariance matrix*

Description

Covariance matrix of parameters if available

Usage

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

Arguments

`object`	The garma_model object
`...`	Other parameters. Ignored.

Value

(double) estimated variance-covariance matrix of the parameter estimates

Index