

# Package ‘garma’

October 21, 2020

**Type** Package

**Title** Fitting and Forecasting Gegenbauer ARMA Time Series Models

**Version** 0.9.6

**Date** 2020-10-09

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

**Description** Methods for estimating univariate long memory-seasonal/cyclical Gegenbauer time series processes. See for example (2018) <doi:10.1214/18-STS649>. Refer to the vignette for details of fitting these processes.

**License** GPL-3

**URL** <https://github.com/rlph50/garma>

**Encoding** UTF-8

**LazyData** true

**Depends** forecast

**Imports** Rsolnp,

ggplot2,  
pracma,  
signal,  
zoo,  
lubridate,  
FKF,  
nloptr,  
crayon,  
utils

**Suggests** longmemo,

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

RoxygenNote 7.1.1  
VignetteBuilder knitr

R topics documented:

AIC.garma_model . . . . .	2
coef.garma_model . . . . .	3
extract_arma . . . . .	3
fitted.garma_model . . . . .	4
forecast.garma_model . . . . .	4
garma . . . . .	5
garma_ggtsdisplay . . . . .	8
ggbr_semipara . . . . .	9
ggplot.garma_model . . . . .	10
gg_raw_pgram . . . . .	10
logLik.garma_model . . . . .	11
plot.garma_model . . . . .	12
predict.garma_model . . . . .	12
print.garma_model . . . . .	13
print.garma_semipara . . . . .	14
print.ggbr_factors . . . . .	14
residuals.garma_model . . . . .	15
summary.garma_model . . . . .	15
vcov.garma_model . . . . .	16
version . . . . .	16
<b>Index</b>	<b>17</b>

---

AIC.garma_model	<i>AIC for model</i>
-----------------	----------------------

---

Description

AIC for model if available.

Usage

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

Arguments

- object           The garma\_model object
- ...             Other parameters. Ignored.

---

coef.garma_model	<i>Model Coefficients</i>
------------------	---------------------------

---

**Description**

Model Coefficients/parameters.

**Usage**

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

**Arguments**

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

---

extract_arma	<i>Extract underlying ARMA process.</i>
--------------	-----------------------------------------

---

**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.

**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)))
```

---

fitted.garma_model	<i>Fitted values</i>
--------------------	----------------------

---

### 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.

---

forecast.garma_model	<i>Forecast future values.</i>
----------------------	--------------------------------

---

### 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, ...)
```

### 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
...	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	<i>garma: A package for estimating and forecasting Gegenbauer time series models.</i>
-------	---------------------------------------------------------------------------------------

---

## 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),
  include.drift = (order[2] > 0),
  method = "Whittle",
  allow_neg_d = FALSE,
  opt_method = NULL,
  m_trunc = 50,
  min_freq = 0,
  max_freq = 0.5,
  control = NULL
)
```

## 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. WARNING: Currently only d==0 or d==1 are allowed.

<code>k</code>	(int) This is the number of (multiplicative) Gegenbauer terms to fit. Note the 'QML' method only allows $k=1$ .
<code>include.mean</code>	(bool) A boolean value indicating whether a mean should be fit. Note that no mean term is fit if the series is integer differenced.
<code>include.drift</code>	(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.
<code>method</code>	(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 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) '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 $\epsilon_t$ are highly skewed and/or have heavy tails (skewed and/or leptokurtic). However the asymptotic theory for the WLL method is not complete and so standard errors are not available for most parameters.
<code>allow_neg_d</code>	(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.
<code>opt_method</code>	(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 $k\geq 2$ . 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: <ul style="list-style-type: none"> <li>• coby algorithm in package nloptr</li> <li>• directL algorithm in package nloptr</li> <li>• BBOptim from package BB</li> <li>• psoptim from package pso</li> <li>• hjkb from dfoptim package</li> <li>• nmkb from dfoptim package</li> <li>• solnp from Rsolnp package</li> <li>• gosolnp from Rsolnp package</li> <li>• 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.</li> </ul> <p>Note further that if you specify a <math>k&gt;1</math>, then inequality constraints are required, and this will further limit the list of supported routines.</p>
<code>m_trunc</code>	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.
control	(list) list of optimisation routine specific values.

## Details

The GARMA model is specified as

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

where

- $\phi(B)$  represents the short-memory Autoregressive component of order p,
- $\theta(B)$  represents the short-memory Moving Average component of order q,
- $(1 - 2u_i B + B^2)^{d_i}$  represents the long-memory Gegenbauer component (there may in general be k of these),
- $X_t$  represents the observed process,
- $\epsilon_t$  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 "arma" 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	<i>ggtsdisplay of underlying ARMA process.</i>
-------------------	------------------------------------------------

---

**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,
  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 Gegenbauer 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 "gamma\_semipara".

**Examples**

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

---

ggplot.garma_model	<i>ggplot of the Forecasts of the model.</i>
--------------------	----------------------------------------------

---

### 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'
ggplot(mdl, h = 24, include_fitted = FALSE, ...)
```

### 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
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)
ggplot(mdl)
```

---

gg_raw_pgram	<i>Display raw periodogram</i>
--------------	--------------------------------

---

### 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)
```

---

logLik.garma_model	<i>Log Likelihood</i>
--------------------	-----------------------

---

**Description**

Log Likelihood, or approximate likelihood or part likelihood, depending on the method.

**Usage**

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

**Arguments**

`object` The garma\_model object

`...` Other parameters. Ignored.

---

plot.garma_model	<i>Plot Forecasts from model.</i>
------------------	-----------------------------------

---

### 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	<i>The predict future values.</i>
---------------------	-----------------------------------

---

### Description

Predict ahead using algorithm of Ferrara, Laurent & Guegan, Dominique, 2001. "Forecasting with k-Factor Gegenbauer Processes: Theory and Applications," Journal of Forecasting, John Wiley & Sons, Ltd., vol. 20(8), pages 581-601, December.

### 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      *print 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 Semiparametric Estimates*

---

**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

---

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.

---

summary.garma\_model *summarise 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)
```

---

<code>vcov.garma_model</code>	<i>Covariance matrix</i>
-------------------------------	--------------------------

---

**Description**

Covariance matrix of parameters if available

**Usage**

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

**Arguments**

<code>object</code>	The <code>garma_model</code> object
<code>...</code>	Other parameters. Ignored.

---

<code>version</code>	<i>garma package version</i>
----------------------	------------------------------

---

**Description**

The `version` function returns the `garma` package version.

**Usage**

```
version()
```

**Examples**

```
library(garma)
garma::version()
```



# Index

AIC.garma\_model, 2

coef.garma\_model, 3

extract\_arma, 3

fitted.garma\_model, 4

forecast.garma\_model, 4

garma, 5

garma\_ggtsdisplay, 8

gg\_raw\_pgram, 10

ggbr\_semipara, 9

ggplot.garma\_model, 10

logLik.garma\_model, 11

plot.garma\_model, 12

predict.garma\_model, 12

print.garma\_model, 13

print.garma\_semipara, 14

print.ggbr\_factors, 14

residuals.garma\_model, 15

summary.garma\_model, 15

vcov.garma\_model, 16

version, 16